PARTICIPANTS	MEMBERS OF SOFTWARE HOUSES, USERS, DEVELOPERS AND IMPLEMENTORS OF COMMUNICATIONS SERVICES AND SYSTEMS, DATA PROCESSING WORKERS.
PREREQUISITES	THE PARTICIPANTS DO NOT NEED ANY EXPERIENCE IN HANDLING OR PROGRAMMING THE PROTOCOL TESTER. BUT THEY SHOULD HAVE FUNDAMENTAL KNOWLEDGE IN COMMUNICATIONS TECHNOLOGY, SYSTEM ARCHITECTURE AND DATA PROCESSING.
GOAL	THE GOAL OF THE COURSE IS TO TEACH THE PARTICIPANTS THE FUNDAMENTAL OPERATING FUNCTIONS OF THE PROTOCOL TESTER K1197, AS WELL AS FUNDAMENTAL CONCEPTS OF THE PROGRAMMING LANGUAGE FORTH.
TEACHING METHODS	LECTURES, PRACTICAL EXERCISES AND DEMONSTRATIONS
DURATION	4 DAYS (4 x 6 HOURS)

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_05470

SUMMARY

- HARDWARE COMPONENTS
- STARTUP
- MENU SYSTEM
- DISK UTILITIES
- PROGRAM USAGE
- K1197 EDITOR

SUMMARY

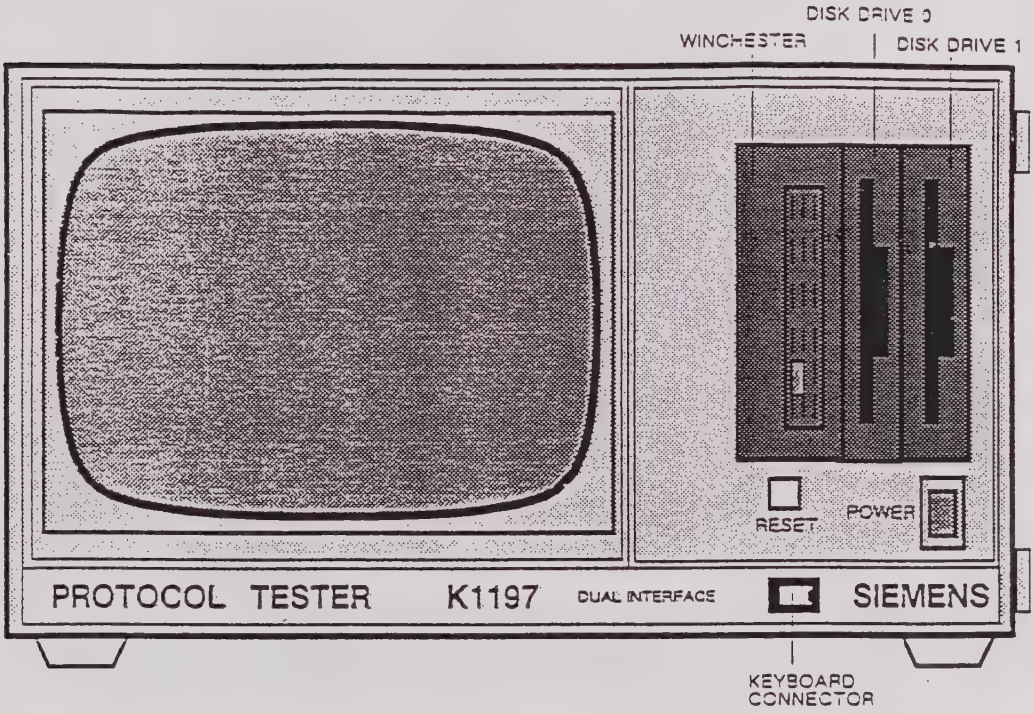
- FORTH INTRODUCTION
- FORTH CONCEPTS
- FORTH DEFINITIONS
- FORTH DATA STRUCTURES
- FORTH CONTROL STRUCTURES
- LAPD TEST PROGRAM INTRODUCTION

HARDWARE COMPONENTS

SUMMARY

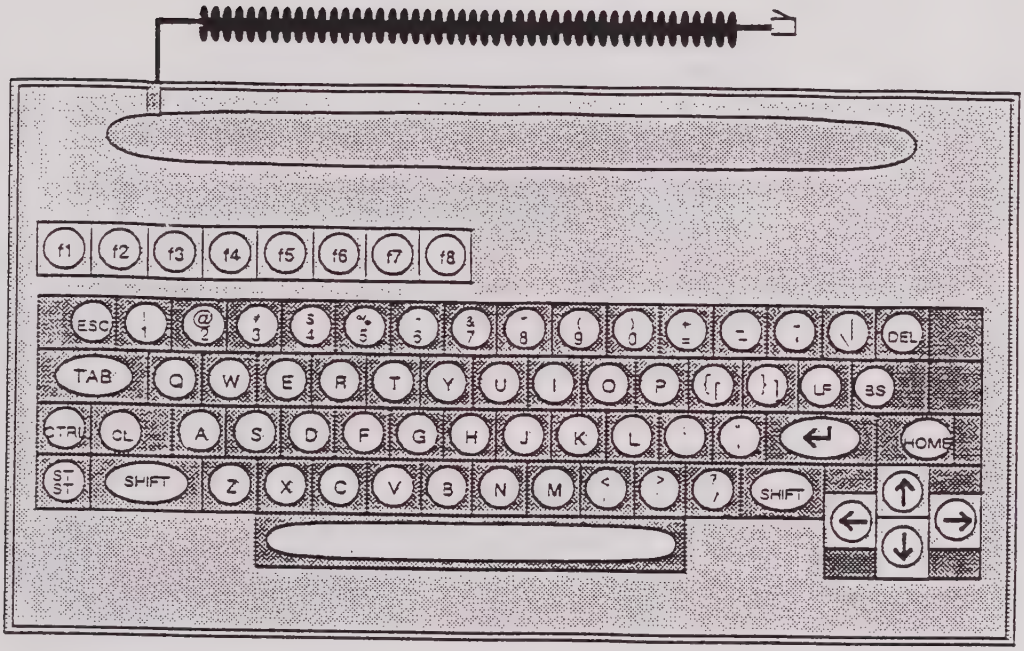
- FRONT VIEW
- KEYBOARD
- SIDE VIEW
- BACK VIEW
- INTERFACE CONNECTOR MODULES
- AUXILIARY CONNECTOR MODULE
- HARDWARE BLOCK DIAGRAM
- MEMORY MAP OF MAIN CPU
- MEMORY MAP OF FECPU
- MEMORY MAP OF BAI2

FRONT VIEW



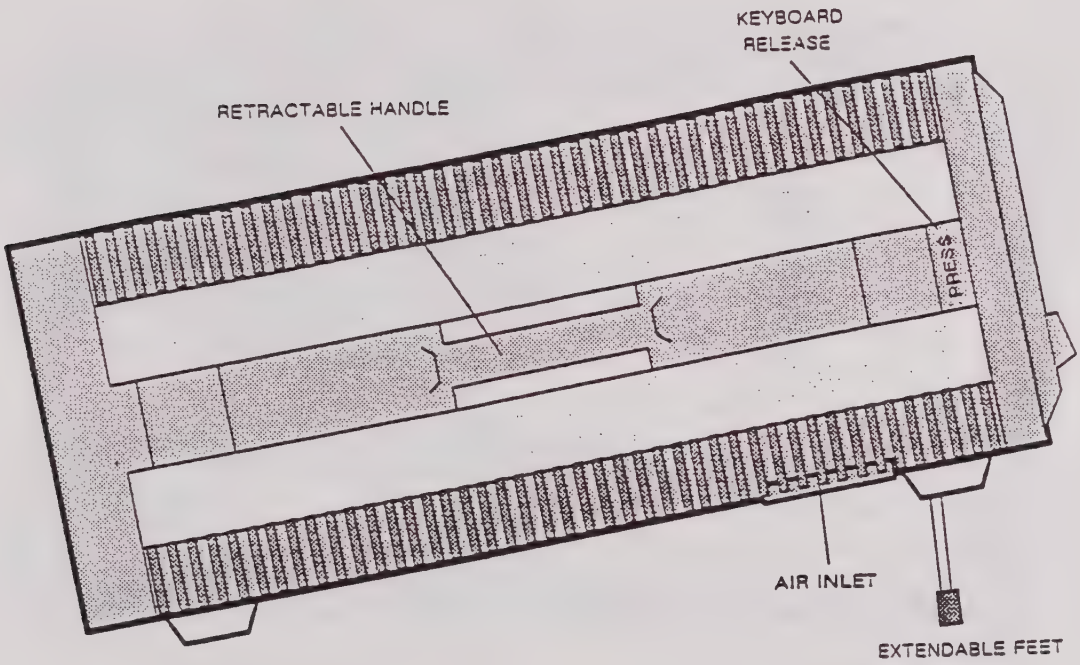
- 40 MBYTES HARDDISK DRIVE (ACCESS TIME < 28 ms)
- DUAL 3.5 INCH FLOPPY DISK DRIVES (1,3 MBYTE/790 KBYTE FORMATTED EACH)
- COLOUR DISPLAY SCREEN
- KEYBOARD CONNECTOR
- RESET BUTTON
- POWER SWITCH

KEYBOARD



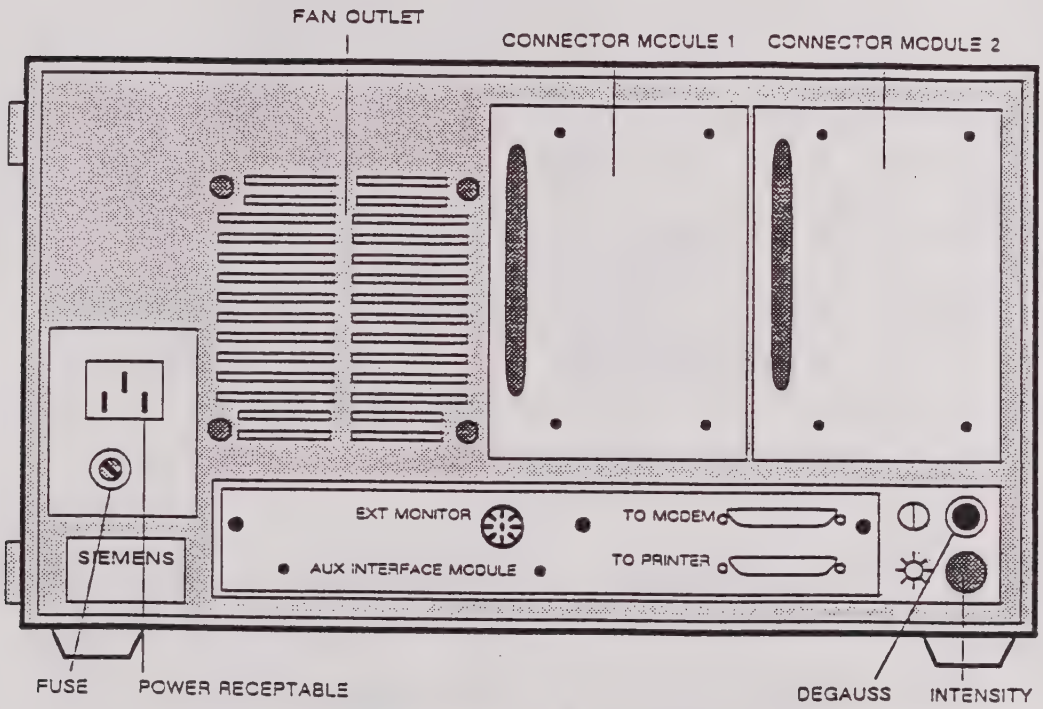
- DETACHABLE FULL ASCII KEYBOARD WITH 8 PROGRAMMABLE FUNCTION KEYS

SIDE VIEW



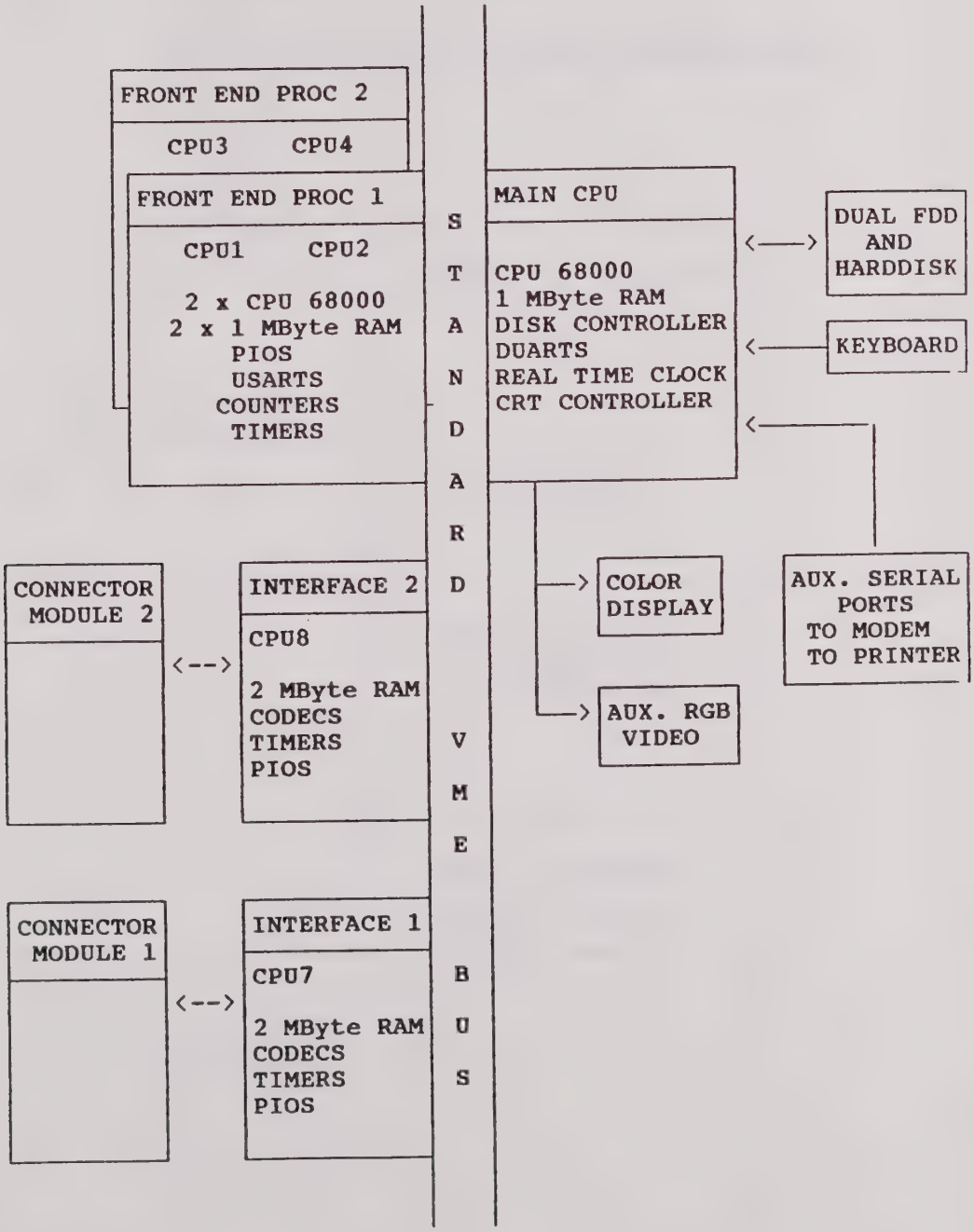
- RETRACTABLE HANDLE
- EXTENDABLE FEET
- KEYBOARD RELEASE BUTTON

BACK VIEW

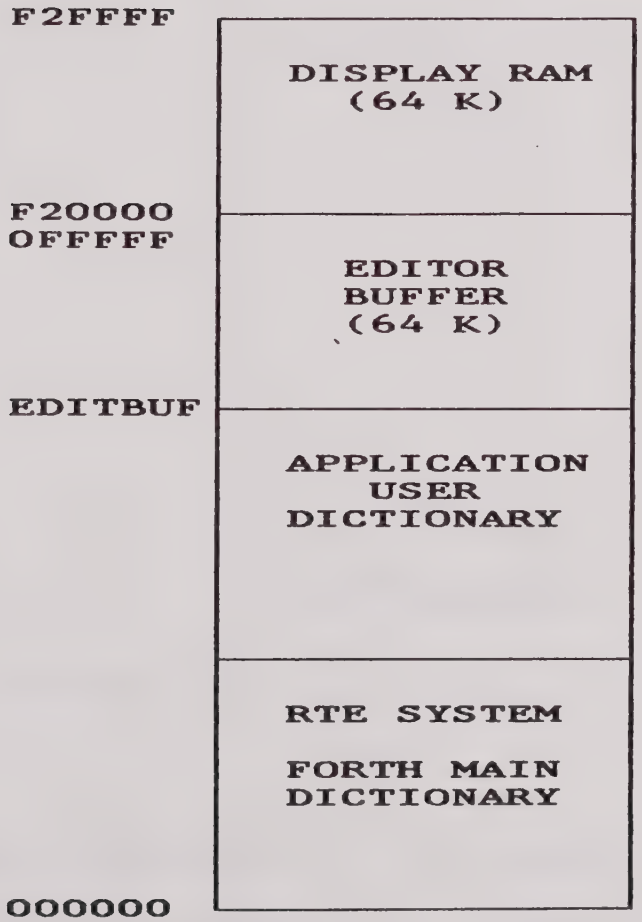


- 2 INTERFACE CONNECTOR MODULES
- AUXILIARY CONNECTOR MODULE
- POWER CONNECTION
- MAIN FUSE
- CRT INTENSITY REGULATOR

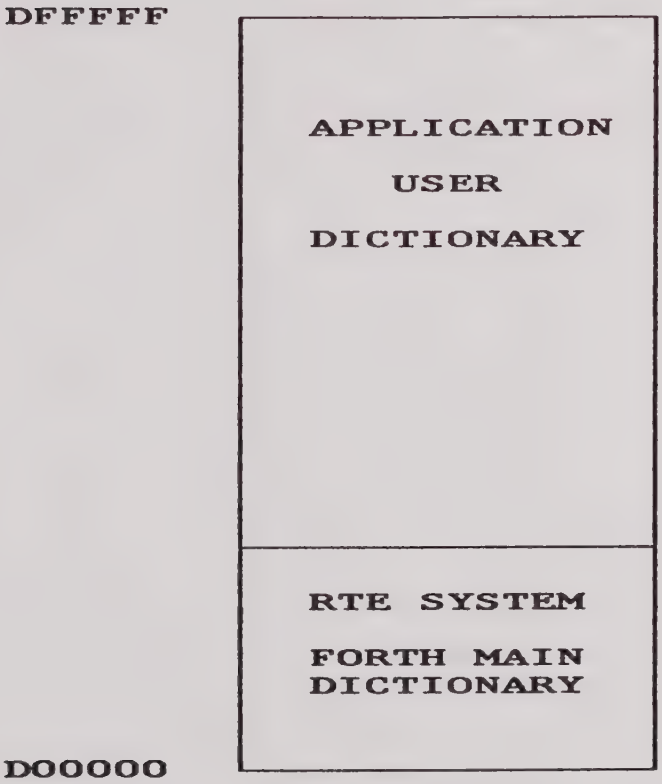
HARDWARE BLOCK DIAGRAM



MEMORY MAP OF MAIN CPU



MEMORY MAP OF FECP (CPU1)



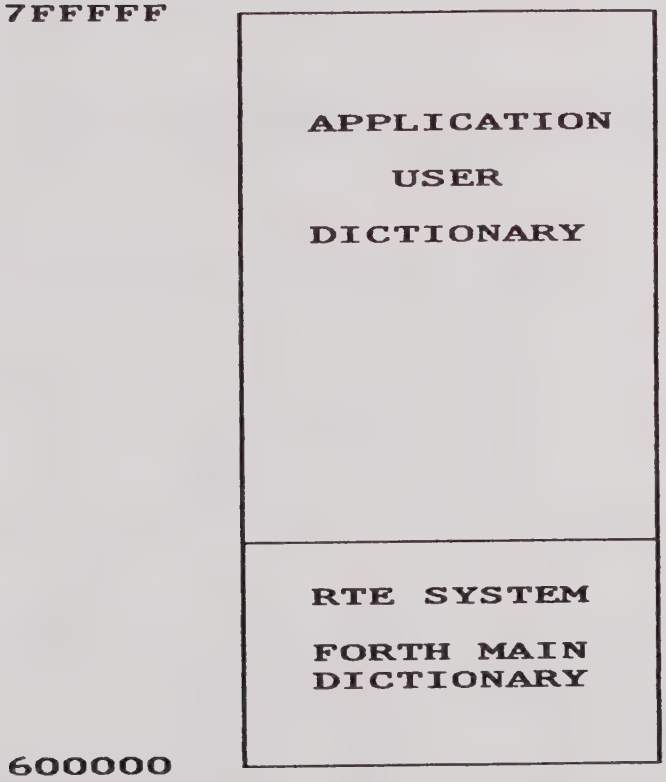
ADDRESSAREAS OF THE OTHER FECP'S:

CPU2: C00000 ... CFFFFF

CPU3: B00000 ... BFFFFF

CPU4: A00000 ... AFFFFF

MEMORY MAP OF BAI2 (CPU7)



ADDRESSAREA OF THE OTHER BAI2:

CPU8: 400000 ... 5FFFFF

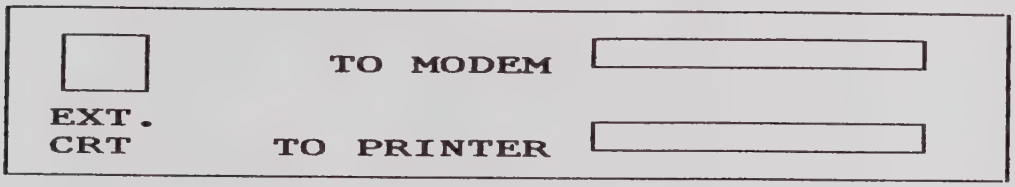
INTERFACE CONNECTOR MODULE

THE PROTOCOL TESTER K1197 MAY BE EQUIPED WITH UP TO 2 INTERFACE MODULES.

THE FOLLOWING INTERFACE MODULES ARE AVAILABLE AND MAY BE COMBINED:

ISDN	ISDN-BAI BASIC ACCESS INTERFACES: **S0** **Up0**
ISDN AND CCS#7	ISDN-PRI PRIMARY RATE INTERFACES: **T1** FOR 1.544 MHZ **CEPT** FOR 2.048 MHZ
V/X	V/X- INTERFACES: **V.24/V.28/RS-232-C** **X.21/V.11** **V.36** **V.35**

AUXILIARY CONNECTOR MODULE



THE AUXILIARY CONNECTOR MODULE CONSISTS OF:

- AN ASYNCHRONOUS SERIAL REMOTE PORT CONNECTOR, TO MODEM (TO DCE), V.24/V.28/RS-232-C/ISO 2110

- AN ASYNCHRONOUS SERIAL PRINTER PORT CONNECTOR, TO PRINTER (TO DTE), V.24/V.28/RS-232-C/ISO 2110

- AN RGB-TTL OUTPUT FOR AN EXTERNAL COLOR MONITOR
  EGA

STARTUP

SUMMARY

- POWER ON
- SELF-DIAGNOSTIC ROUTINES
- BOOT SEQUENCE
- LOADING FORTH EXTENSIONS

POWER ON

INSERT A SYSTEM DISK INTO DISK DRIVE 0 OR 1.

THE LABEL SHOULD FACE THE CRT DISPLAY.

THE K1197 TRIES TO BOOT FROM WINCHESTER IF NO DISK IS INSERTED IN THE DISK DRIVE(S).

TURN THE POWER SWITCH ON OR PRESS THE RESET BUTTON IF THE UNIT IS ALREADY ON

THE CRT TAKES APPROXIMATELY 5 SECONDS TO WARM UP.

THEN THE DIAGNOSTICS AND BOOT MESSAGES APPEAR.

SELF-DIAGNOSTIC ROUTINES

AN EXTENSIVE **SELF-DIAGNOSTIC** PROGRAM IS CONTAINED IN THE SYSTEM'S **BOOT PROM** AND IS **AUTOMATICALLY EXECUTED** WHENEVER THE POWER IS TURNED ON OR THE K1197's RESET BUTTON IS PRESSED.

THIS PROGRAM CAUSES THE FOLLOWING ACTIONS:

MAIN CPU

INITIALIZE THE CRT

ROM CHECK --> DISPLAY: "ROM OK"

LOAD ALL FECPUs WITH TEST PROGRAM

START ALL CORRECTLY LOADED FECPUs

RAM TESTS --> DISPLAY: "1 MB RAM OK"

WRITE/READ AND INTERRUPT TESTS --> DISPLAY: "PERIPHERALS OK"

SYNCHRONIZATION WITH FECPUs

HALT ALL FECPUs

SELF-DIAGNOSTIC ROUTINES

THE FOLLOWING ACTIONS ARE PERFORMED FOR EACH FECPU.

FECPUs

RAM TESTS

WRITE/READ TESTS AND TESTS WITH INTERRUPTS

SYNCHRONIZATION WITH MAIN CPU

STOP

DURING THESE TESTS, THE SYSTEM MAY DISPLAY ERROR MESSAGES IN RED BACKGROUND WHICH ARE IN ALL CASES SELF EXPLANATORY

BOOT SEQUENCE

ONCE THE SELF-DIAGNOSTICS ARE SUCCESSFULLY COMPLETED, THE MAIN CPU PERFORMS THE FOLLOWING SEQUENCE TO LOAD THE OPERATING SYSTEM:

- CHECK IF THERE IS AN **RTE SYSTEM** DISK **IN FLOPPY DRIVE DR0 OR DR1** OR IF AN RTE SYSTEM HAS BEEN INSTALLED ON **WINCHESTER PARTION WD0.**

  IF NOT, THE USER IS PROMPTED TO **INSERT AN RTE SYSTEM DISK** IN FLOPPY DRIVE DR0.

- ALL " *.sys " FILES ARE LOADED.

- IF THE FILE " main.sys " IS MISSING, THE USER IS ASKED TO **REPLACE THE DISK** BY ONE WHICH CONTAINS A COMPLETE OPERATING SYSTEM.

BOOT SEQUENCE

- FOR EACH FILE A MESSAGE OF THE FOLLOWING FORM IS DISPLAYED.

  LOADING "xxx.sys" AT <LOCATION>.

  WHERE

"xxx.sys"	IS THE NAME OF THE FILE BEING LOADED AND
<LOCATION>	IS THE ADDRESS IN MEMORY (HEXADECIMAL) WHERE THE PROGRAM IS LOADED.

```
                SIEMENS  K1197    Bootsstrap 1.0

MAIN CPU DIAGNOSTICS: ROM OK, 1 MB RAM OK, PERIPHERALS OK


INTERFACE 1   APPLICATION PROCESSORS
FECP1  CPU A  OPERATING SYSTEM: LOADED
FECP1  CPU B  LOADING  "fecp1B.sys"  AT 0XD0000
INTERFACE

INTERFACE 2   APPLICATION PROCESSORS
FECP2  CPU A
FECP2  CPU B
INTERFACE


Harddisk drive is NEC 40 Mb
BOOTING FROM FLOPPY 0
```

BOOT SEQUENCE

ERROR MESSAGES

- IF A FLOPPY DISK **ERROR** APPEARS DURING BOOTING IT WILL BE SHOWN IN **GREEN AND RED LETTERS.**

- ANY DISK ERROR DETECTED WILL CAUSE THE **LAST DISK OPERATION** TO BE RETRIED UNTIL THE OPERATION IS ERROR FREE.

- AFTER **10 UNSUCCESSFUL RETRIES** THE ENTIRE BOOT SEQUENCE IS STARTED FROM THE **BEGINNING.**

NOTE: THE OCCURENCE OF ERRORS DURING BOOTING USUALLY INDICATES **WORN OUT STORAGE MEDIUM** WHICH SHOULD BE REPLACED.

LOADING FORTH EXTENSIONS

AFTER SUCCESSFULLY LOADING THE SYSTEM FILES THE SYSTEM WILL **INITIALIZE AND START-UP FORTH.**

THE BINARY-FILE " **ALOAD**" IS LOADED ON THE **MAIN CPU**, AND THE " **ALOADn**" FILES ARE LOADED ON THE CORRESPONDING **FECPUs.**

THE **MENU SYSTEM**, WHICH IS STORED IN THE ALOAD FILE AS OBJECT CODE **IS LOADED AUTOMATICALLY.**

THE MENU SYSTEM IS **STARTED** BY THE **FORTH WORD ALOAD**, WHICH IS EXECUTED **AUTOMATICALLY** AFTER SUCCESSFULLY LOADING THE ALOAD FILE.

IF THE **ALOAD(n) FILE** FOR ANY CPU IS **NOT AVAILABLE** ON DISK OR THE FORTH WORD ALOAD IS NOT DEFINED THE USER WILL GET THE MESSAGE:

" FORTH entry point not found"

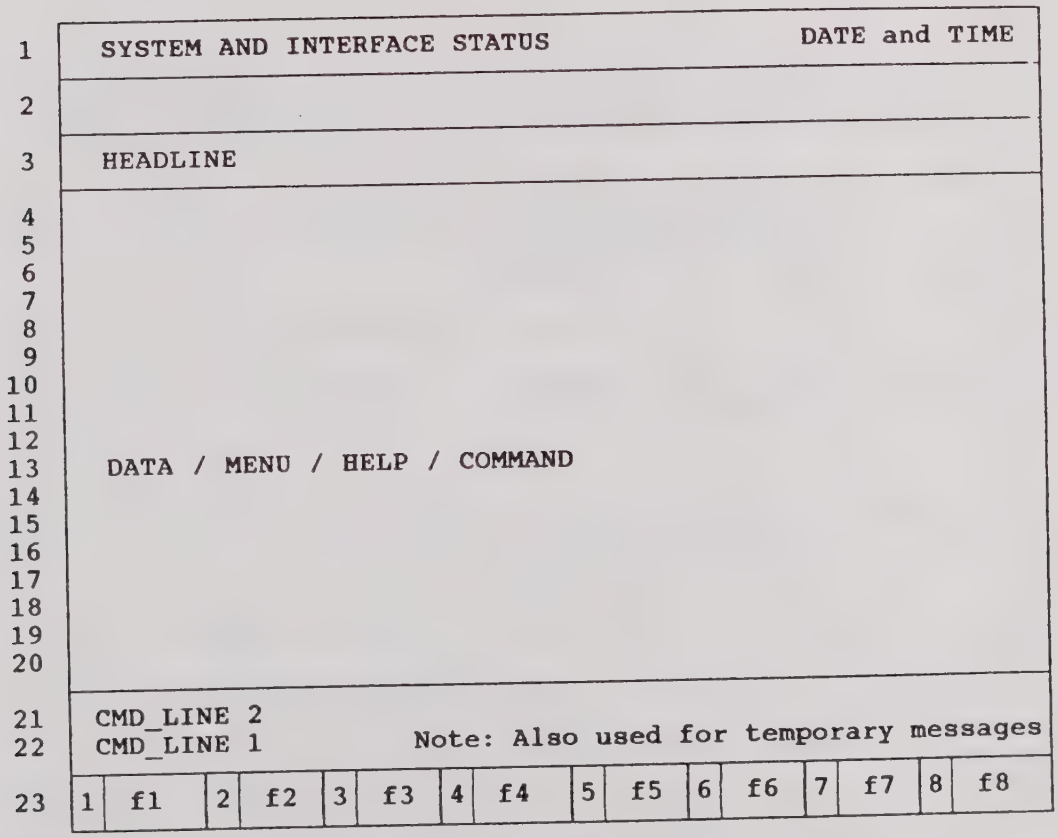
AND THE FORTH INTERPRETER IS RUNNING ON THIS CPU, WITHOUT RUNNING THE MENU SYSTEM.

MENU SYSTEM

SUMMARY

- SCREEN DISPLAY LAYOUT
- FUNCTION KEYS
- COMMAND EXECUTION
- SPECIAL FUNCTIONS
- MENU ICONS
- WALKING THROUGH THE MENU SYSTEM
- MENU WINDOWS

SCREEN DISPLAY LAYOUT



SCREEN DISPLAY LAYOUT

LINE No	NAME	CONTENT/DESCRIPTION
1 – 2	STATUS LINES	PROVIDE SET UP AND CONFIGURATION INFORMATION e.g. CONNECTED INTERFACE
3	HEAD-LINE	HEADLINE FOR THE DISPLAY AREA
4 – 20	DISPLAY	GENERAL DATA AND MENU WINDOW DISPLAY AREA
13 – 22	COMMAND LINES	FOR DIRECT COMMAND ENTRY (INSTEAD OF FUNCTION KEYS)
22	MESSAGE	USED FOR TEMPORARY MESSAGES
23	f(1-8)	FUNCTION KEYS (f1-f8)

FUNCTION KEYS

THERE IS A SET OF FUNCTION KEYS AVAILABLE ON THE BOTTOM LINE OF THE DISPLAY.

THE COMMON KEYS FOR ALL APPLICATION PROGRAMS ARE THE FOLLOWING ONES:

QUIT_MENU	LEAVES THE MENU SYSTEM CONTROL.
MENU	BACK TO MENU SYSTEM CONTROL. DISPLAYS THE MENU WINDOW **UP TO THE LAST ACTIVE MENU** WHICH WAS OPENED BEFORE QUIT_MENU.
HELP	DISPLAYS THE CORRESPONDING HELP WINDOW.
QUIT_HELP	CLOSES THE HELP WINDOW AND RETURNS TO THE PREVIOUSLY OPENED MENU WINDOW.

FUNCTION KEYS

COMMAND GIVES CONTROL TO THE FORTH WINDOW WHICH ENABLES THE USE OF KEYBOARD FOR TYPING COMMANDS.

QUIT_COM RETURN CONTROL BACK TO THE MENU SYSTEM. (YOU CAN USE **CURSOR KEY** --> INSTEAD OF QUIT_COM, WHICH ADDITIONALLY DISPLAYS THE PREVIOUSLY ACTIVE MENU WINDOW.)

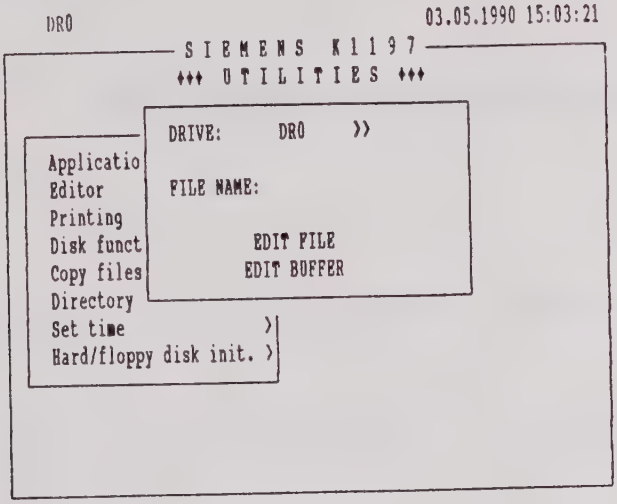
BY USING COMMAND (f3) THE MENU SYSTEM WINDOW REMAINS STILL VISIBLE AND THE LINES 21-22 CAN BE USED FOR TYPING FORTH COMMANDS.

THE COMMAND WINDOW CAN BE EXTENDED UP TO 10 COMMAND LINES.

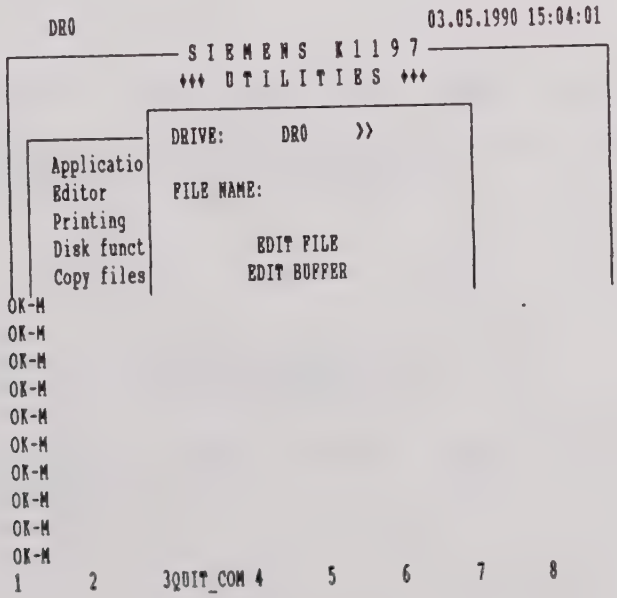
TO EXTEND THE COMMAND WINDOW BY 1 LINE YOU HAVE TO PRESS THE COMBINATION:

TO REDUCE THE COMMAND WINDOW BY 1 LINE YOU HAVE TO PRESS THE COMBINATION:

EXAMPLE



AFTER PRESSING f3 AND PRESSING 8 TIMES THE COMBINATION CTRL --> THERE ARE 10 COMMAND LINES AVAILABLE.



COMMAND EXECUTION

THERE ARE TWO DIFFERENT WAYS TO EXECUTE A COMMAND:

- SELECT THE FUNCTION IN THE MENU WINDOW OR PRESS ONE OF THE FUNCTION KEYS

- GO TO THE COMMAND MODE (PRESS f3) AND ENTER THE NAME OF THE COMMAND AT THE KEYBOARD

EXAMPLE:

TO INVOKE THE EDITOR YOU CAN EITHER

OPEN THE EDITOR MENU AND EXECUTE THE COMMAND **EDIT FILE**

OR

PRESS **COMMAND**

AND TYPE THE WORD **EDIT** FOLLOWED BY RETURN.

SPECIAL FUNCTIONS

SHIFT + CTRL + f8

RETURNS THE SYSTEM TO THE FORTH INTERPRETER IN CASE OF A LOCK-UP OR PROGRAM ERROR. THE MENU SYSTEM IS LEFT.

SHIFT + CTRL + f1

PRINTS A HARDCOPY OF THE CURRENT SCREEN DISPLAY.

DEFAULT PRINTER CONFIGURATION:

1200 BAUD, 8 BITS/CHARACTER, ASCII, DTR AND XON/OFF CONTROL,
LF CR END OF LINE,
80 CHARACTERS/LINE, 60 LINES/PAGE

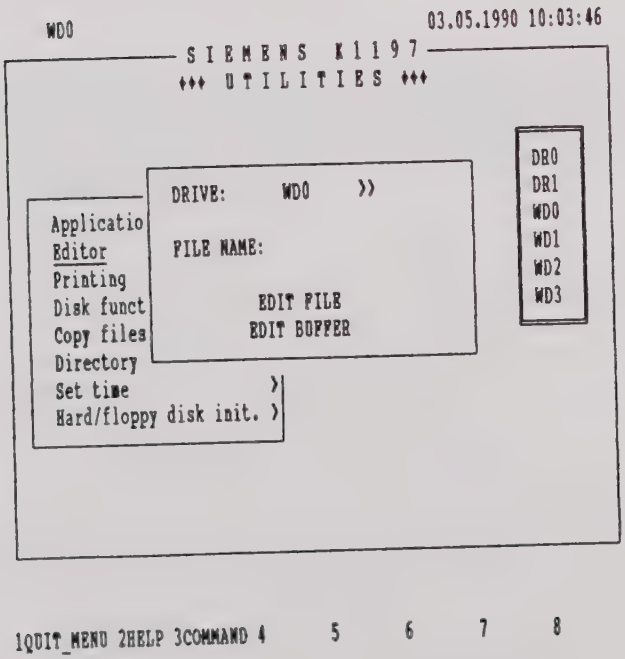
THE PRINTER CONFIGURATION MAY BE CHANGED IN THE **PRINTER CONFIGURATION MENU.**

MENU ICONS

THE MENU SYSTEM IS STRUCTURED IN WINDOWS.

THE BORDERS OF THE MENU ARE EITHER SINGLE LINE OR DOUBLE LINE BORDERS.

DOUBLE LINE BORDERS ARE USED TO INDICATE THE LAST MENU IN A CHAIN.



MENU ICONS

TO THE RIGHT OF A SELECTABLE ITEM, ONE OF THE FOLLOWING ARROWS MAY APPEAR:

> OR »

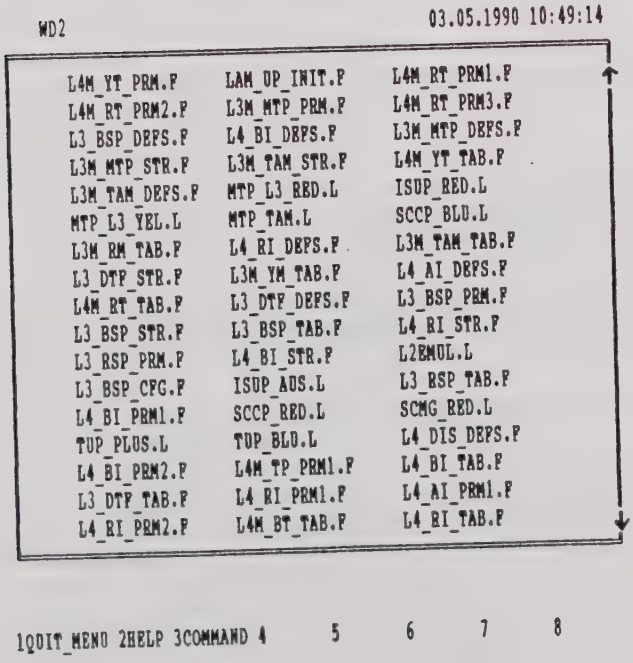
THE **SINGLE ARROW** INDICATES THAT THERE IS **A SUBMENU AVAILABLE** FOR THIS ITEM WHICH WILL BE DISPLAYED WHEN THE **CURSOR RIGHT** KEY IS PRESSED.

THE **DOUBLE ARROW** INDICATES THAT THE **FIELD TAKES A NUMBER OF OPTIONS** WHICH MAY BE SCROLLED THROUGH PRESSING THE SPACE BAR OR LISTED USING THE CURSOR RIGHT KEY.

MENU ICONS

AT THE TOP AND BOTTOM OF THE RIGHT HAND BORDER OF A MENU THERE WILL APPEAR ARROWS IF THE MENU CONTAINS MORE LINES THAN CAN BE DISPLAYED IN THIS WINDOW.

THE CURSOR KEYS ↑ (UP) AND ↓ (DOWN) CAN BE USED IN THAT CASE TO SCROLL THROUGH ALL AVAILABLE ITEMS.



WALKING THROUGH THE MENU SYSTEM

WITHIN A MENU, THE FOLLOWING KEYS MAY BE USED TO MOVE AND TO PERFORM ACTIONS:

CURSOR KEYS

NORMAL

↑ MOVES TO **NEXT** SELECTABLE **ITEM ABOVE** THIS ONE.

↓ MOVES TO **NEXT** SELECTABLE **ITEM BELOW** THIS ONE.

<-- MOVES TO THE **PREVIOUS MENU**
NOTE: SOME MENUS DO NOT ALLOW THIS KEY.

IN STRING AND NUMERIC ENTRY FIELDS: MOVE **CURSOR** ONE CHARACTER **BACKWARD.**

--> MOVES TO THE **NEXT SUBMENU** IF ONE EXISTS.

IN STRING AND NUMERIC ENTRY FIELDS: MOVE **CURSOR** ONE CHARACTER **FORWARD.**

WALKING THROUGH THE MENU SYSTEM

CURSOR KEYS

SHIFTED

THE SHIFTED CURSOR KEYS ARE USED ONLY IN CONNECTION WITH *THE STRING AND NUMERIC ENTRY FIELDS*.

<- **DELETE THE CHARACTER** UNDER THE CURSOR AND SHIFT THE FOLLOWING CHARACTERS ONE STEP BACKWARD.

-> **SHIFT THE CHARACTERS** FROM THIS PLACE ON **ONE STEP FORWARD** AND CREATE A BLANK BETWEEN THE ALREADY EXISTING CHARACTERS.

WALKING THROUGH THE MENU SYSTEM

CURSOR KEYS

CONTROLLED

↑ **MOVE UP ONE PAGE** ON SCROLLING MENUS.

↓ **MOVE DOWN ONE PAGE** ON SCROLLING MENUS.

<-- *IN STRING AND NUMERIC ENTRY FIELDS:* **MOVE THE CURSOR TO THE FIRST CHARACTER** IN THE STRING.

--> *IN STRING AND NUMERIC ENTRY FIELDS:* **MOVE THE CURSOR TO THE LAST CHARACTER** IN THE STRING.

WALKING THROUGH THE MENU SYSTEM

CURSOR KEYS

SHIFTED + CONTROLLED

↑ MOVES TO FIRST ITEM IN MENU.

↓ MOVES TO LAST ITEM IN MENU.

<- RETURNS TO THE FIRST MENU.

-> RETURNS TO THE PREVIOUSLY OPENED MENU.

WALKING THROUGH THE MENU SYSTEM

HOME KEY

- MOVE TO FIRST ITEM IN MENU (SAME AS SHIFT-CTRL-↑).

BACK SPACE (BS), DELETE (DEL)

- *IN STRING AND NUMERIC ENTRY FIELDS:* ERASE THE CHARACTER BEFORE THE CURSOR.

RETURN

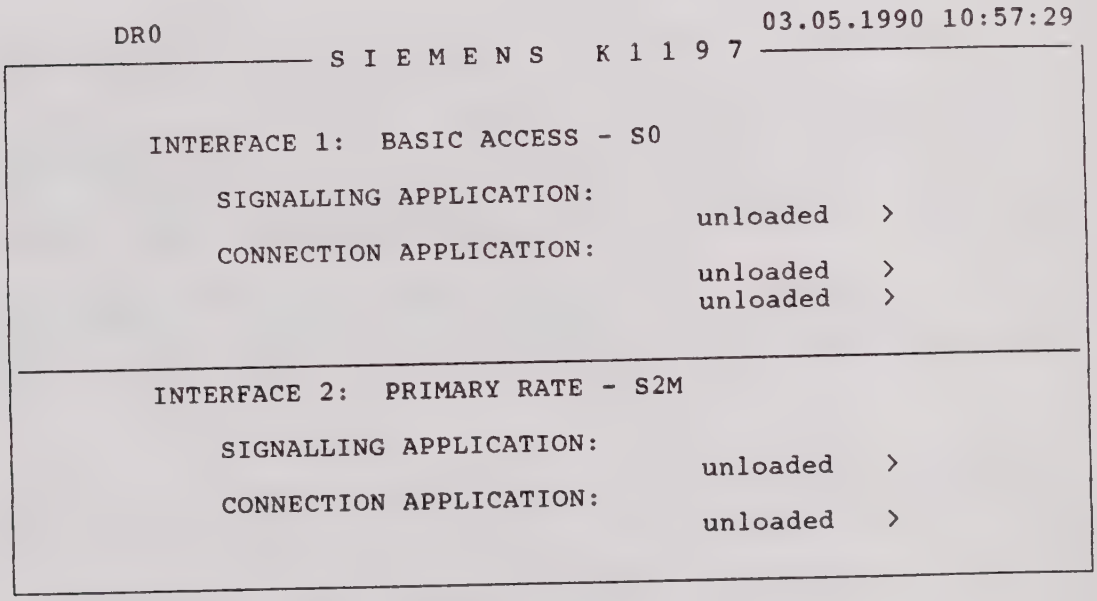
- SELECT AN ITEM, EXECUTE A COMMAND OR TERMINATE INPUT OF A VALUE.

SPACE BAR

- WHEN THE CURSOR IS ON A FIELD WHICH TAKES A NUMBER OF OPTIONS, THE SPACE BAR MAY BE USED TO SCROLL THROUGH THE AVAILABLE OPTIONS.

MENU WINDOWS

APPLICATION MENU



THE MAIN APPLICATION MENU IS THE FIRST ACTIVE MENU WHICH APPEARS WHEN THE K1197 IS SWITCHED ON.

IN THIS MENU THE INSTALLED INTERFACE AND THE NAMES OF LOADED APPLICATION PROGRAMS ARE DISPLAYED.

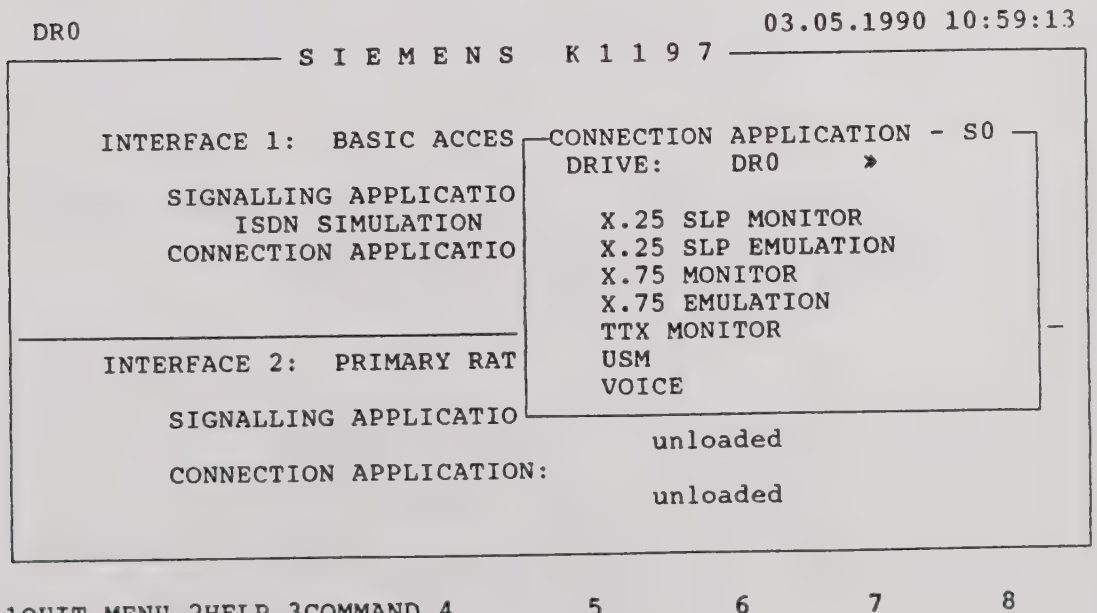
MENU WINDOWS

APPLICATION MENU

APPLICATION PROGRAMS CAN BE LOADED AT ANY TIME.

TO **GET A LIST OF POSSIBLE APPLICATION PROGRAMS** WHICH CAN BE LOADED, POSITION THE CURSOR AT THE COORESPONDING SIGNALLING OR CONNECTION APPLICATION AND **PRESS THE CURSOR RIGHT** KEY.

TO **LOAD** ONE OF THE POSSIBLE APPLICATION PROGRAMS MOVE THE CURSOR UP OR DOWN UNTIL THE **DESIRED PROGRAM IS HIGHLIGHTED** AND THEN **PRESS THE RETURN KEY.**



MENU WINDOWS

THE **CURRENT LOADING SITUATION** CAN BE SAVED TO / LOADED FROM A SETUP-FILE USING THE FOLLOWING **FUNCTION KEYS:**

SAVE CONF OPEN A MENU FOR SAVING CURRENTLY LOADED APPLICATION PROGRAMS SO THAT THEY CAN BE RELOADED LATER USING **LOAD CONF.**

SAVES ALSO THE CURRENT PRINTER CONFIGURATION.

```
DRIVE:        DR0      >>

FILE NAME:  SETUP.C
LOADED BY SYSTEM BOOT: YES >>

      SAVE CONFIGURATION
```

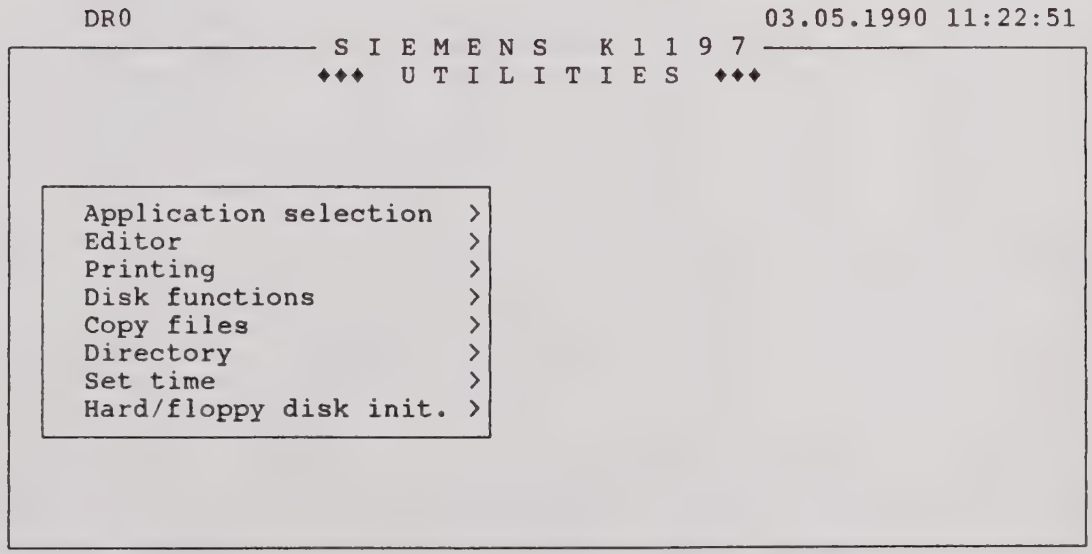
LOAD CONF OPEN A MENU FOR LOADING A PREVIOUSLY SAVED APPLICATION PROGRAM CONFIGURATION AND RESTORING THE PRINTER CONFIGURATION.

```
DRIVE:        DR0      >>

FILE NAME:  SETUP.C


      LOAD CONFIGURATION
```

MENU WINDOWS

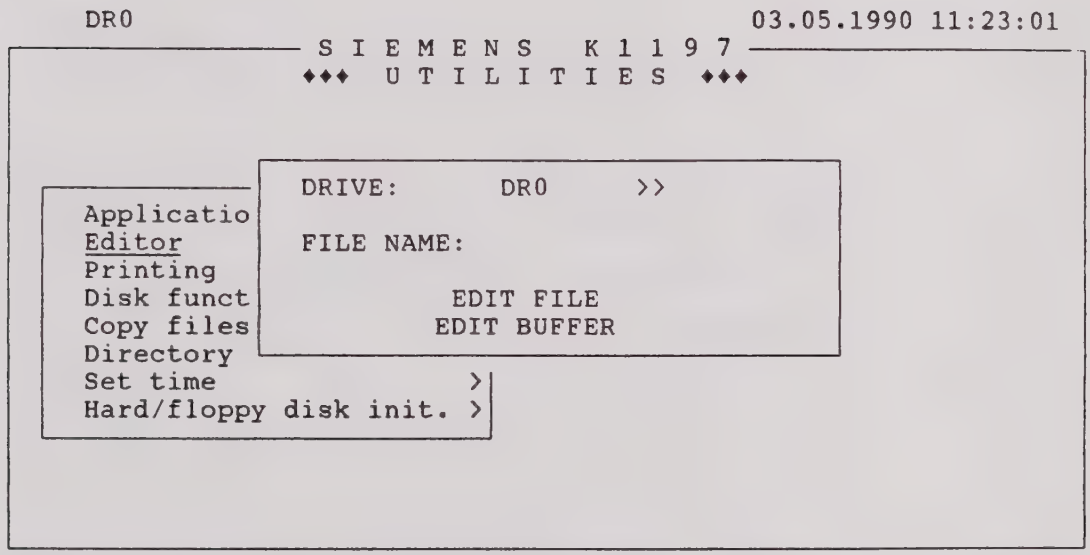
THE APPLICATION MENU IS LEFT FOR THE MAIN UTILITIES MENU BY PRESSING CURSOR LEFT.

MAIN UTILITIES MENU



MENU WINDOWS

EDITOR MENU



DRIVE: : SELECT DISK DRIVE

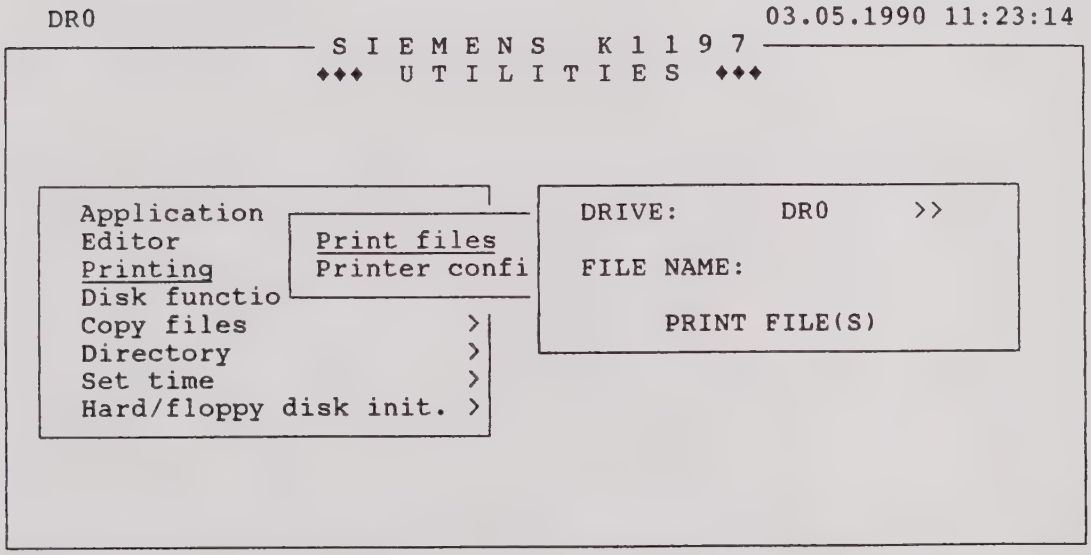
FILE NAME: : ENTER FILE NAME

EDIT FILE: : READ ABOVE ENTERED FILE INTO **EDITBUF** FOR EDITING

EDIT BUFFER : DISPLAY THE CONTENTS OF **EDITBUF** FOR EDITING

MENU WINDOWS

PRINT FILES MENU



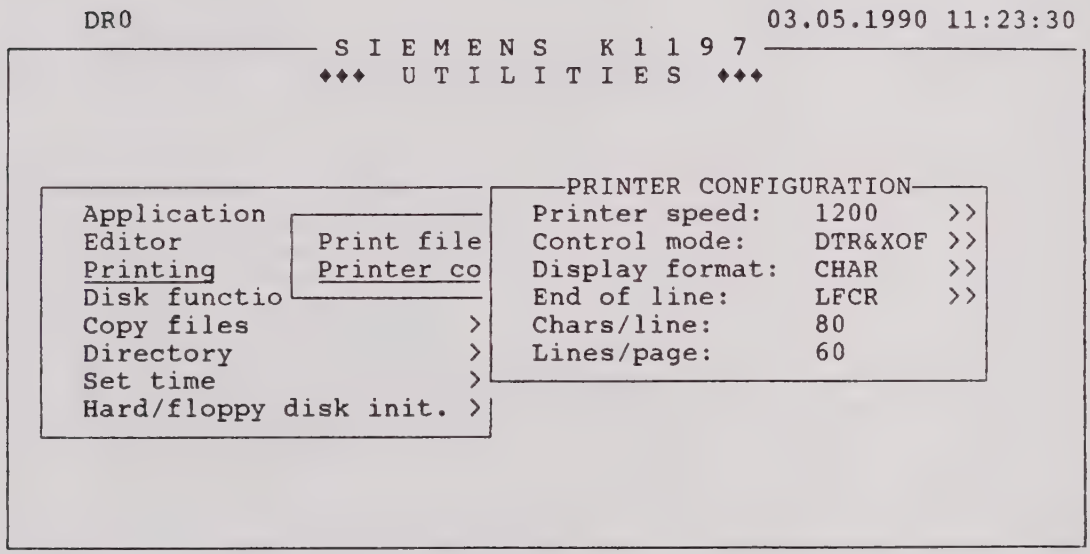
DRIVE: : SELECT DISK DRIVE

FILE NAME: : ENTER FILE NAME

PRINT FILE: : EXECUTE FILE PRINTING. THE FILES TO BE PRINTED ARE ALSO DISPLAYED WHEN PRINTING.

MENU WINDOWS

PRINTER CONFIGURATION MENU



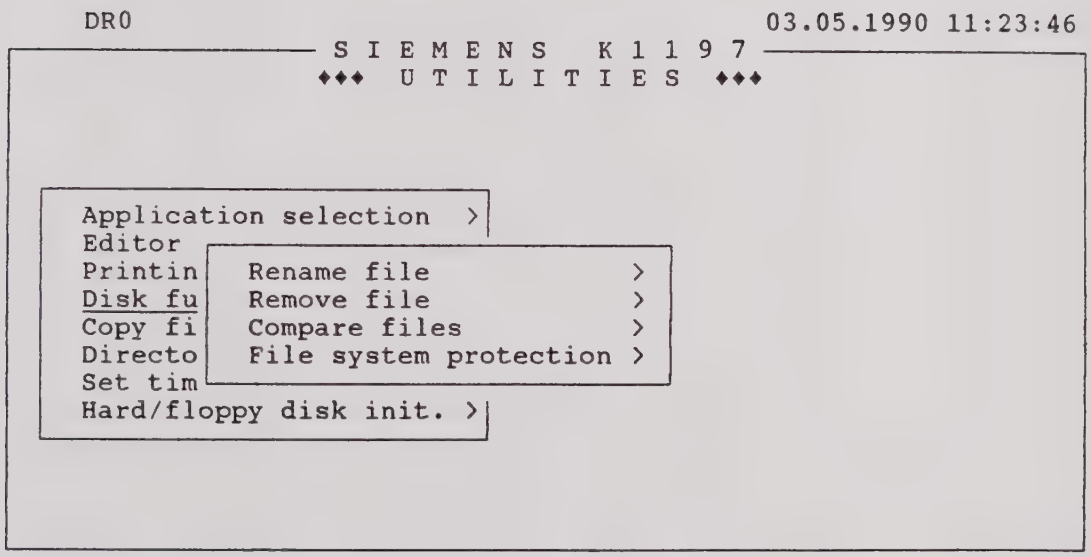
1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

SET THE ITEMS OF THE PRINTER CONFIGURATION:

FOR EACH ITEM YOU HAVE TO **SCROLL** THROUGH THE EXISTING ALTERNATIVES **BY PRESSING THE SPACE BAR** UNTIL THE DESIRED VALUE IS FOUND **OR BY OPENING THE COMPLETE SELECTION MENU** (CURSOR RIGHT) AND MAKING THE SELECTION THERE (RETURN).

MENU WINDOWS

DISK FUNCTIONS



RENAME FILE : SELECT MENU TO RENAME A FILE

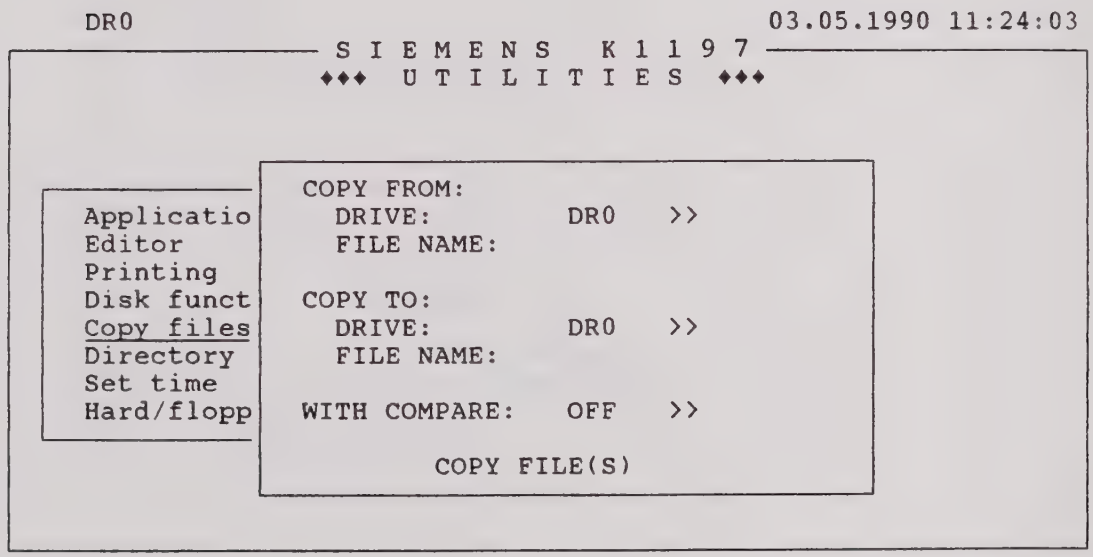
REMOVE FILE : SELECT MENU TO REMOVE FILES FROM THE DIRECTORY

COMPARE FILES : SELECT MENU TO COMPARE FILES

FILE SYSTEM PROTECTION : SELECT MENU TO RE(SET) FILE SYSTEM WRITE PROTECTION

MENU WINDOWS

COPY FILES MENU



1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

DRIVE: : SELECT DISK DRIVE

COPY FROM: : ENTER SOURCE FILE NAME

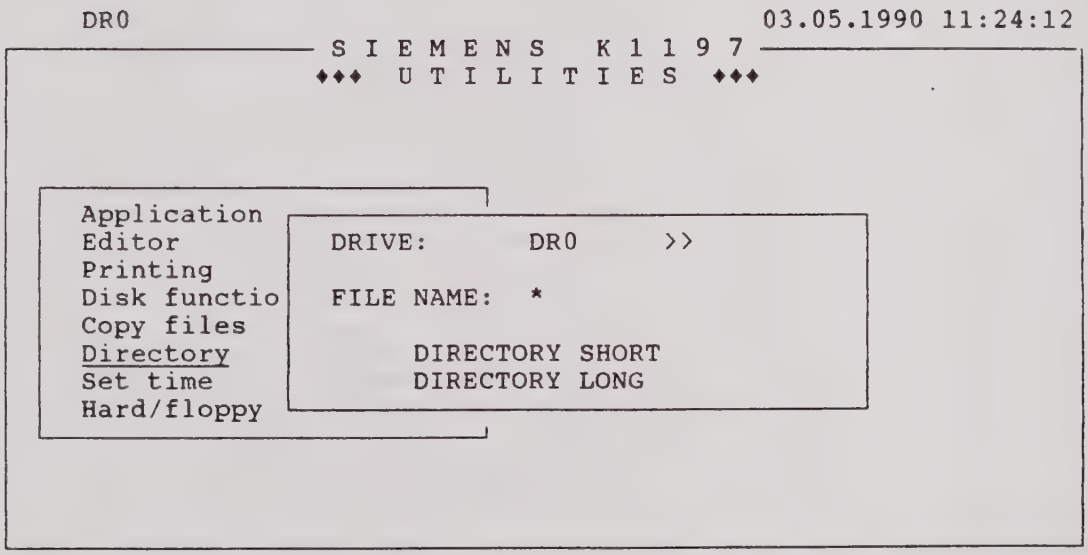
COPY TO: : ENTER DESTINATION FILE NAME

WITH COMPARE: : WHEN SELECTED, COMPARE IS EXECUTED AFTER COPYING

COPY FILE(S) : EXECUTE FILE COPYING

MENU WINDOWS

DIRECTORY MENU



DRIVE: : SELECT DISK DRIVE

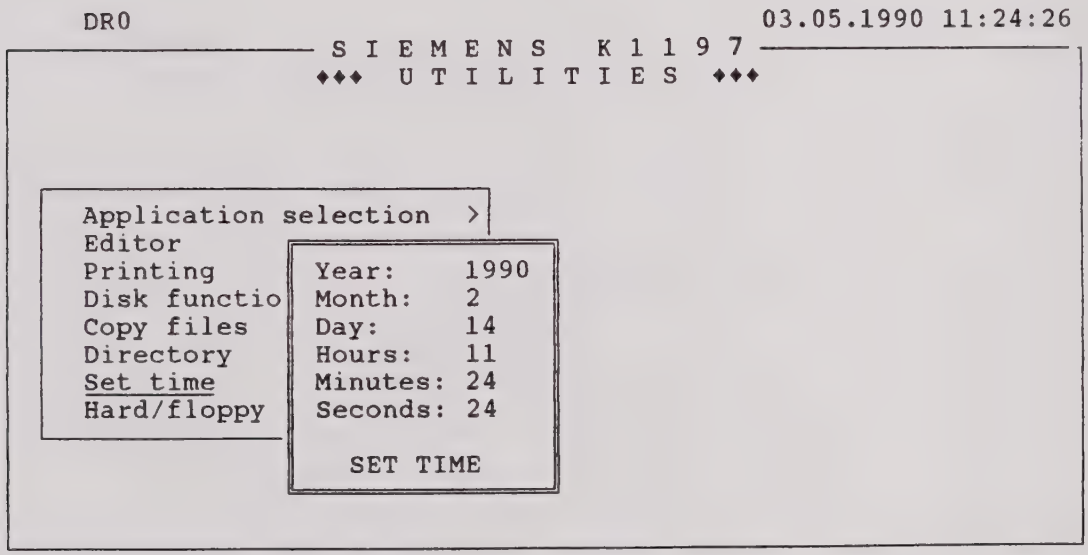
FILE NAME: : ENTER FILE NAME

DIRECTORY SHORT : DISPLAY THE CONTENTS OF THE SELECTED DISK DRIVE IN SHORT FORMAT.

DIRECTORY LONG : DISPLAY THE CONTENTS OF THE SELECTED DISK DRIVE IN LONG FORMAT.

MENU WINDOWS

SET TIME MENU

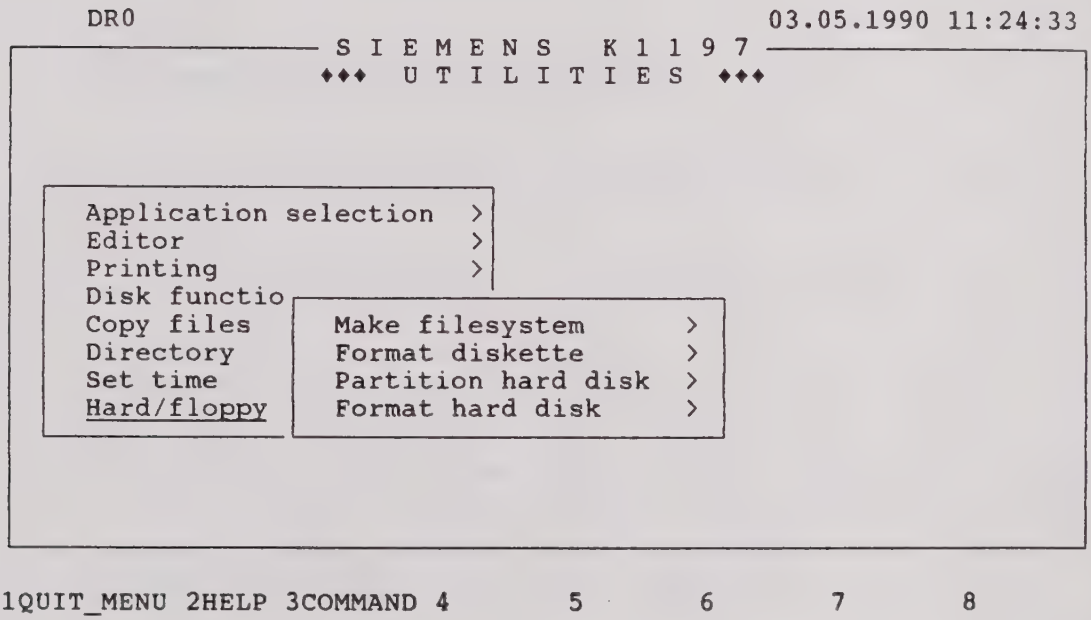


THE TIME IS ENTERED AS NUMBERS IN THE CORRESPONDING FIELDS.

TO EXECUTE THE TIME SETTING MOVE THE CURSOR TO **SET TIME** AND PRESS RETURN.

MENU WINDOWS

HARD/FLOPPY DISK INITIALIZATION



MAKE FILESYSTEM : SELECT MENU TO CREATE A FILESYSTEM

FORMAT DISKETTE : SELECT MENU TO FORMAT A FLOPPY DISK

PARTITION HARD DISK : SELECT MENU TO DIVIDE THE WINCHESTER INTO LOGICAL HARD DISK DRIVES

FORMAT HARD DISK : SELECT MENU TO FORMAT AND PARTITION THE HARD DISK

EXERCISE 1

1. Power up the K1197 (or reset it if power is already up).

2. Insert the exercise disk into drive 1.

3. Go to the MAIN UTILITIES MENU.

4. Get a long directory list of the disk in drive 1.

5. Go back to the MAIN UTILITIES MENU.

6. Open the FORTH interpreter window and extend it by 6 lines.

7. Get a short directory list using the FORTH command " *" DIR.

8. Leave the FORTH interpreter window.

EXERCISE 2

1. Change the printer configuration to:

Printer speed:	9600
Control mode:	DTR&XOF
Display format:	RAW
End of line:	LFCR
Chars/line:	80
Lines/page:	60

2. Save the printer configuration on WDO using the **SAVE CONF** command. The filename should be SETUP.C and the file should be loaded automatically when the System is rebooted.

3. Reset the system and see whether the printer configuration has been changed as required in 1.

SOLUTION 1

1. Turn the power switch on or press the **RESET** button if the power is already up.

2. Insert the exercise disk into the right floppy disk drive.

3. The MAIN APPLICATION MENU which appears at POWERUP is left for the MAIN UTILITIES MENU by pressing **cursor left.**

SOLUTION 1 cont'd

4.

4.1. Press the **cursor down** key until **'DIRECTORY'** is highlighted.

4.2. Press the **cursor right** key to select the **DIRECTORY MENU.**

4.3. Move the cursor up so that it points to the field **'DRIVE:'.**

4.4. To select DR1 you can either press the space bar until DR1 is found or hit cursor right to display all available drives then select DR1 and press RETURN.

4.5. Move the cursor to the field **'FILE NAME:'**and type a **'*'**, so that all files will be displayed.

4.6. Execute **DIRECTORY LONG.**

SOLUTION 1 cont'd

5. Go back to the **MAIN UTILITIES MENU** by pressing the **cursor left** key twice.

6. Press the function key **F3** (COMMAND) to open the FORTH window.

   Press the combination **CTRL + Cursor right** 6 times to extend the command window by 6 lines.

7. Get a short directory list by typing the command **" *" DIR** followed by RETURN.

8. Press the function key **F3** (QUIT_COM) or the **cursor right** key to close the FORTH window.

SOLUTION 2

1.

1.1. Select the **PRINTER CONFIGURATION MENU.**

1.2. Move the cursor to the field **'Printer Speed'** and press the Space bar until 9600 is selected.

1.3 Move the cursor to the next field and make your selection etc.

1.4 Go back to the **MAIN UTILITIES MENU** by pressing the cursor left key twice.

SOLUTION 2 cont'd

2.

2.1. Press function key **F6** (SAVE CONF).

2.2 Move the cursor up until the field '**DRIVE:**' is high-lighted.
Press the space bar until WDO is selected.

2.3 Move the cursor down so that it points to the field '**FILE NAME:**' and type the name **SETUP.C**.

2.4 Move the cursor down to the field '**LOADED BY SYSTEM BOOT**' and select **YES**.

2.5 Execute **SAVE CONFIGURATION**.

2.6 Press function key **F6** (QUIT_CONF).

3. Reset the system and select the **PRINTER CONFIGURATION MENU**. The configuration must be the one you selected in 1.

DISK UTILITIES

SUMMARY

- ORGANIZATION OF A HARD/FLOPPY DISK
- FILENAME ENTRY
- FILENAME FORMATS
- WILDCARDS
- DIRECTORY DISPLAY FORMAT (LONG)
- DISK FUNCTIONS
- HARD/FLOPPY DISK INITIALIZATION

ORGANIZATION OF A HARD/FLOPPY DISK

HARD DISK DRIVE

THE K1197 HAS A **40 MBYTE** WINCHESTER DISK DRIVE INSTALLED.

THE PHYSICAL WINCHESTER DRIVE CAN BE DIVIDED INTO A NUMBER **(1 TO 8)** OF LOGICAL PARTITIONS OR **'LOGICAL DISKS'**.

THE NAMES USED FOR THE LOGICAL DISKS ARE:
WD0, WD1, WD2, WD3, WD4, WD5, WD6, WD7

IF, FOR EXAMPLE, THE PHYSICAL DISK IS DIVIDED INTO 4 PARTITIONS, ONLY WD0 THROUGH WD3 ARE VALID.

THE **SIZE** OF THE LOGICAL DISKS IS GIVEN IN BLOCKS (ONE BLOCK = 512 BYTES).

16 BLOCKS OF EACH LOGICAL DISK ARE RESERVED FOR THE DIRECTORY.

ORGANIZATION OF A HARD/FLOPPY DISK

FLOPPY DISK DRIVES

THE K1197 HAS TWO 3.5" FLOPPY DISK DRIVES INSTALLED.

THE TOTAL FORMATTED STORAGE CAPABILITY PER DRIVE IS:

1,3 MBYTE FOR A HIGH DENSITY DISK AND
790 KBYTE FOR A DOUBLE DENSITY DISK

THE NAMES USED FOR THE FLOPPY DISK DRIVES ARE:

DR0 FOR THE LEFT HAND DRIVE AND
DR1 FOR THE RIGHT HAND DRIVE.

10 BLOCKS ARE RESERVED FOR THE DIRECTORY.

FILENAME ENTRY

FILENAMES MAY BE PRECEDED WITH THE DRIVE DESIGNATION **DR0:** OR **DR1:**
OR **WD0:** ... **WD7:**

EXAMPLE: DR0:SOURCE1

FILENAME FORMATS

VALID FILENAME FORMATS ARE:

LONG_FILE_NAME	(UP TO 14 CHARACTERS)
1234	(NUMERIC)
TEST_1.SRC	(OPTIONAL EXTENSION)
FILE_$!%(&"...	(SPECIAL CHARACTERS)

WILDCARDS

WILDCARD ENTRY PERMITS THE USE OF THE FOLLOWING META CHARACTERS:

*	ANY CHARACTER SEQUENCE OR NO CHARACTERS
?	ANY SINGLE CHARACTER
[,]	DELIMITERS TO ALLOW A CHOICE OF CHARACTERS OR A RANGE OF CHARACTERS

EXAMPLES:

[123]	CHOICE OF 1, 2 OR 3
[1-5]	RANGE OF 1 TO 5
[A-Z]	RANGE OF A TO Z

WILDCARDS

FILENAMES MAY CONTAIN WILDCARDS FOR THE FOLLOWING OPERATIONS:

DIRECTORY SHORT
DIRECTORY LONG
REMOVE
PRINT
COPY

YOU SHOULD TAKE CARE IN USING WILDCARDS WITH THE **REMOVE** OR **COPY** OPERATION.

WITH THE COPY OPERATION A **DESTINATION FILENAME** MAY BE ENTERED FOR A SINGLE FILE ONLY. IF MULTIPLE FILES ARE COPIED, THE DESTINATION FILES WILL HAVE THE SAME NAMES AS THE SOURCE FILES.

THE FOLLOWING EXAMPLES SHOW TYPICAL USAGE OF WILDCARDS:

```
DR0:*

DR0:FILE1

DR1:FILE[123]

DR1:FILE?.SRC
```

DIRECTORY DISPLAY FORMAT (LONG)

```
DR0     DIRECTORY LONG:    " *"  FOUND  0 FILES
Volume: K1197_DISK         Size: 1600   Free: 1590

FILENAME       TYPE  SIZE    ATTR.     DATE     TIME
```

TYPE:

DAT	DATA RECORDING FILE
CONF	CONFIGURATION FILE
SRC	SOURCE CODE FILE
OBJ	OBJECT CODE FILE (BINARY)
EXEC	SYSTEM FILE

SIZE:

NUMBER OF BLOCKS (512 BYTES EACH)

ATTRIBUTE:

PERM	PERMANENT
TEMP	TEMPORARY

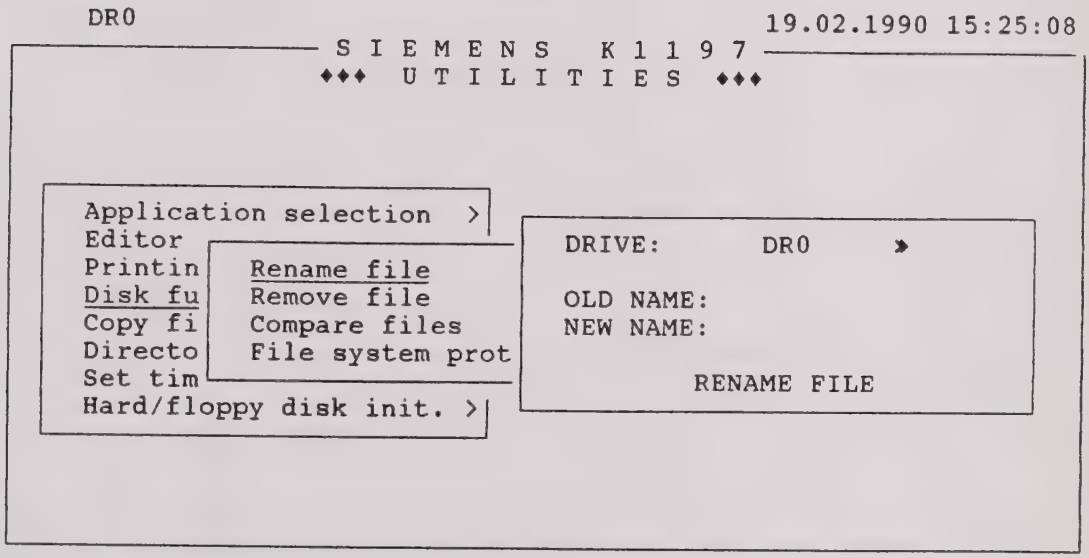
DIRECTORY DISPLAY FORMAT (LONG)

EXAMPLE:

```
DR0                                                    19.02.1990 14:34:27
      DR0  DIRECTORY LONG:  " *"  Found  13 files.
      Volume: EXERCISE-DISK Size: 1600   Free: 1305
      ED-EX-TEXT3     Src   21     Perm  1990-01-19 14:01:39
      ED-EX-TEXT1     Src   23     Perm  1990-01-19 14:01:54
      ED-EX-1         Src   17     Perm  1990-01-19 14:02:08
      SCAN.F          Src   41     Perm  1990-01-19 14:02:24
      ED-ZZZ          Src   1      Perm  1990-01-19 14:02:38
      DECOMPRESS.F    Src   49     Perm  1990-01-19 14:02:54
      ED-EX-3         Src   15     Perm  1990-01-19 14:03:08
      ED-XXX          Src   1      Perm  1990-01-19 14:03:21
      DEMO.SRC        Src   27     Perm  1990-01-19 14:03:36
      SCANM.F         Src   42     Perm  1990-01-19 14:03:52
      ED-EX-2         Src   15     Perm  1990-01-19 14:04:07
      ROMAN.SRC       Src   9      Perm  1990-01-19 14:04:20
      ODERSO.F        Src   24     Perm  1990-01-19 14:04:35


1QUIT_MENU 2HELP 3COMMAND 4       5       6       7       8
```

DISK FUNCTIONS

RENAME FILE MENU



1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

DRIVE: : SELECT DISK DRIVE

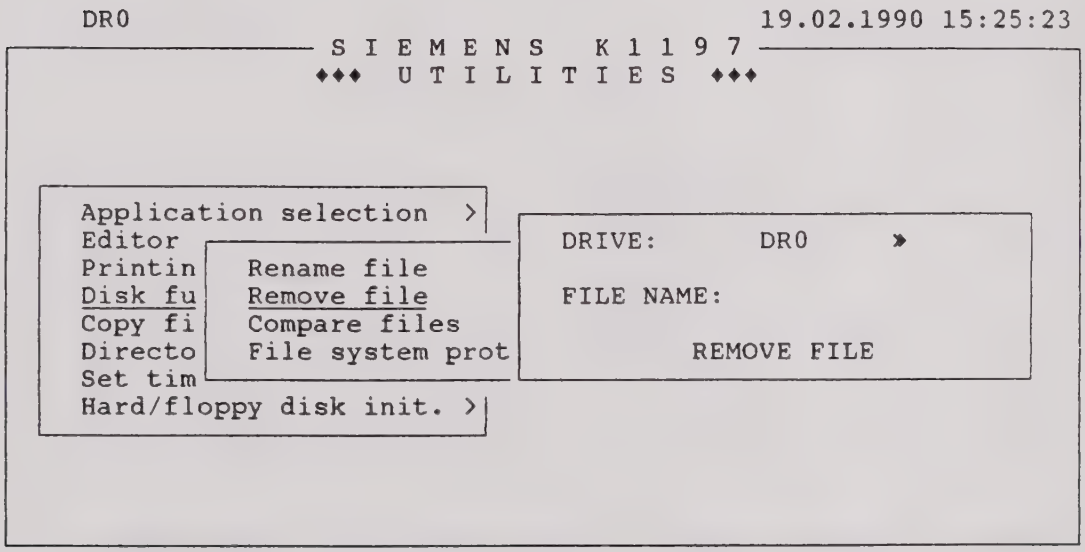
OLD NAME: : ENTER SOURCE FILE NAME

NEW NAME: : ENTER DESTINATION FILE NAME

RENAME FILE : EXECUTE RENAMING

DISK FUNCTIONS

REMOVE FILE MENU



1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

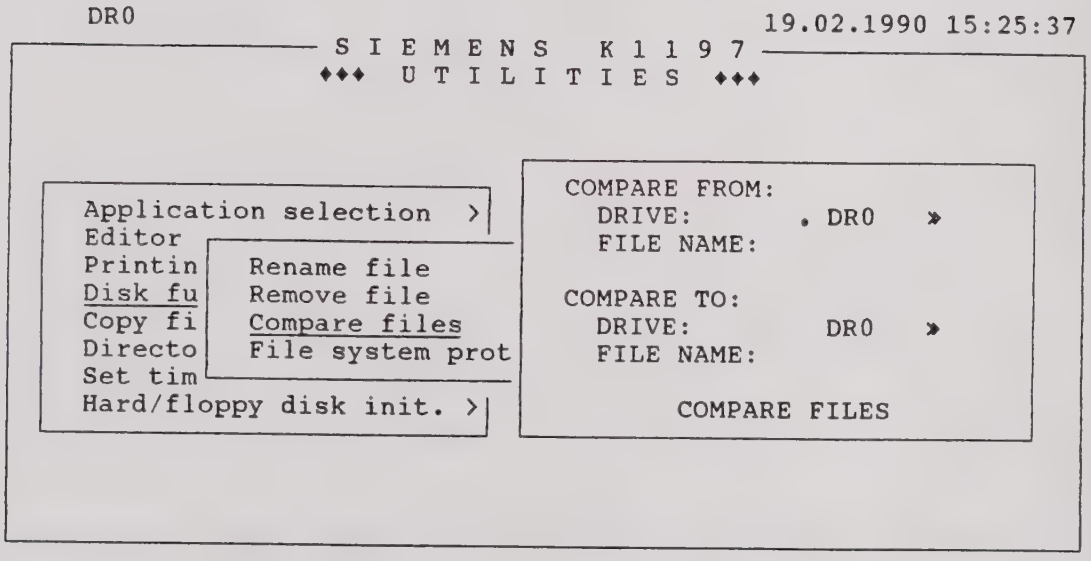
DRIVE: : SELECT DISK DRIVE

FILE NAME: : ENTER FILE NAME

REMOVE FILE : DISPLAY ALL FILE NAMES THAT ARE TO BE REMOVED AND REMOVE THEM WHEN POSITIVELY ACKNOWLEDGED

DISK FUNCTIONS

COMPARE FILES MENU



DRIVE: : SELECT DISK DRIVE

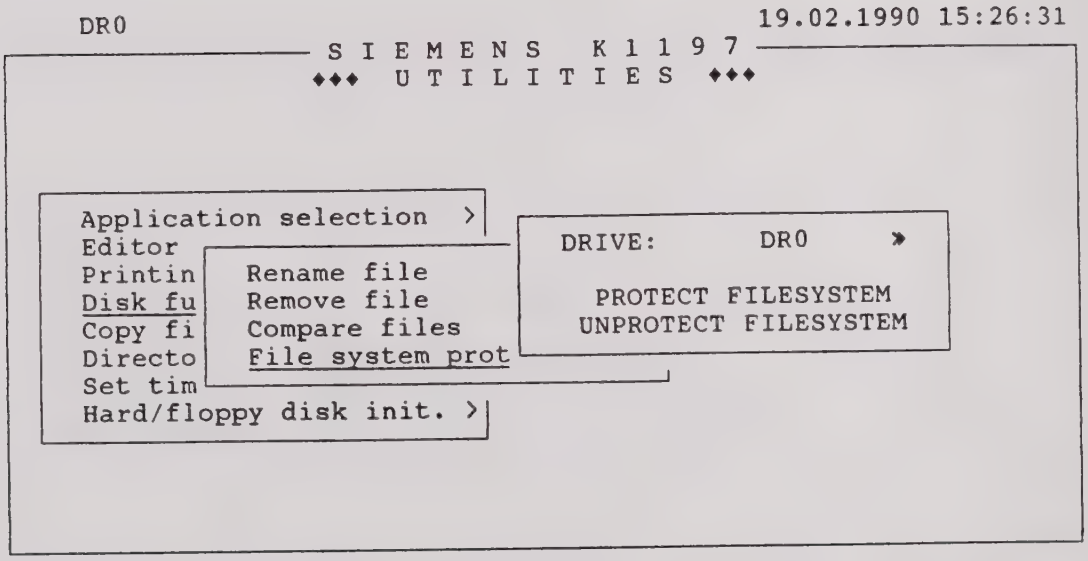
COMPARE FROM: : ENTER SOURCE FILE NAME

COMPARE TO: : ENTER DESTINATION FILE NAME

COMPARE FILES : EXECUTE FILE COMPARE. SHOWS WHETHER OR NOT THE FILES ARE IDENTICAL.

DISK FUNCTIONS

FILESYSTEM PROTECTION MENU



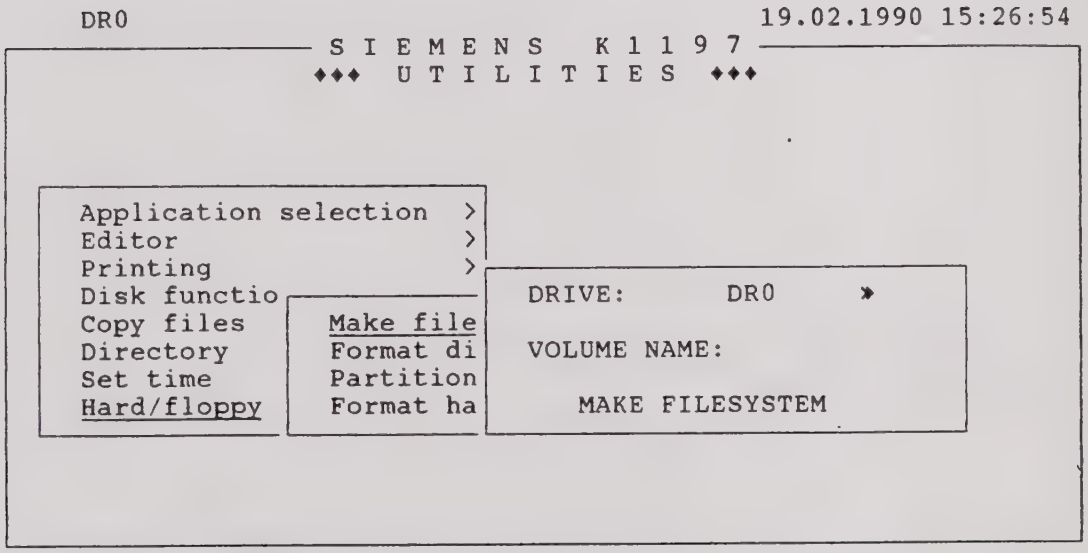
DRIVE: : SELECT DISK DRIVE

PROTECT FILESYSTEM : SET WRITE PROTECTION TO THE FILESYSTEM

UNPROTECT FILESYSTEM : REMOVE WRITE PROTECTION FROM THE FILESYSTEM

HARD/FLOPPY DISK INITIALIZATION

MAKE FILESYSTEM MENU



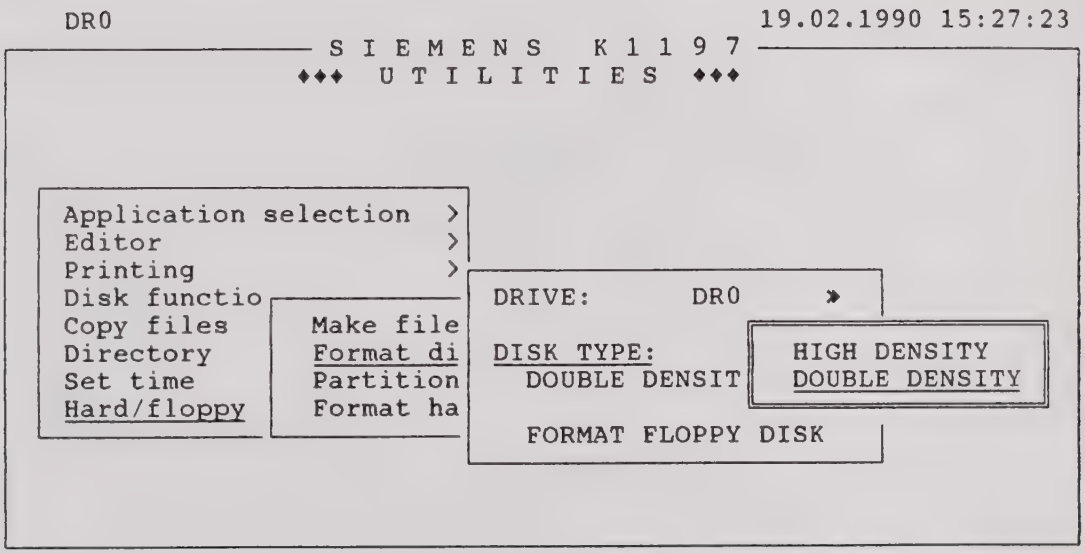
DRIVE: : SELECT DISK DRIVE

VOLUME NAME: : ENTER NAME FOR THE FILESYSTEM

MAKE FILESYSTEM : A WARNING IS GIVEN THAT ALL FILES WILL BE LOST ON THE SELECTED DISK DRIVE. ACKNOWLEDGEMENT IS REQUIRED BEFORE MAKING A FILESYSTEM.

HARD/FLOPPY DISK INITIALIZATION

FORMAT FLOPPY DISK MENU



1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

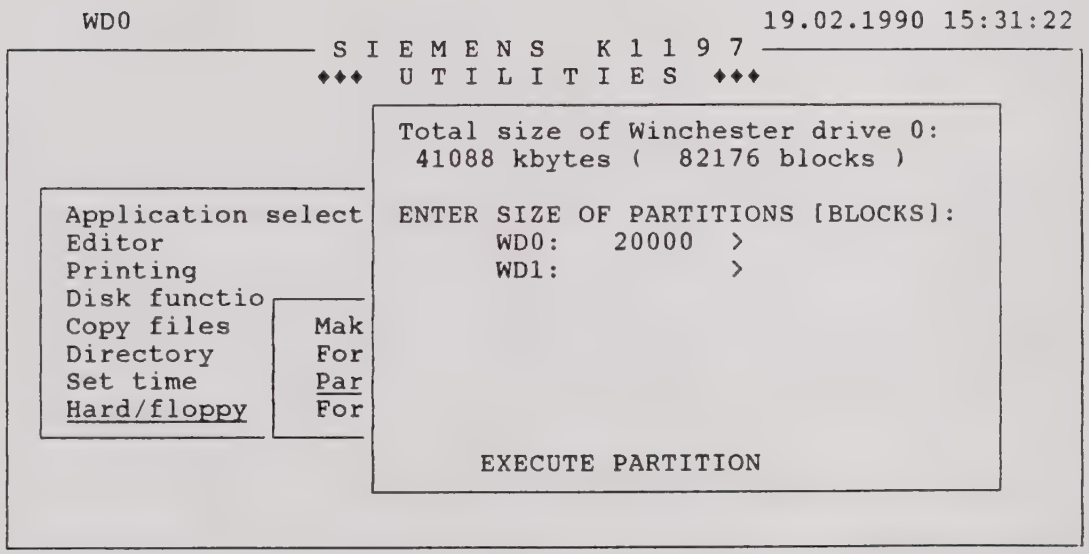
DRIVE: : SELECT DISK DRIVE.

DISK TYPE: : SELECT DISK TYPE.

FORMAT FLOPPY DISK : A WARNING IS GIVEN THAT **ALL FILES WILL BE LOST ON THE SELECTED DISK DRIVE.** ACKNOWLEDGEMENT IS REQUIRED BEFORE FORMATTING THE DISK. (THE DEFAULT VOLUME NAME IS K1197_DISK).

HARD/FLOPPY DISK INITIALIZATION

PARTITION HARD DISK MENU

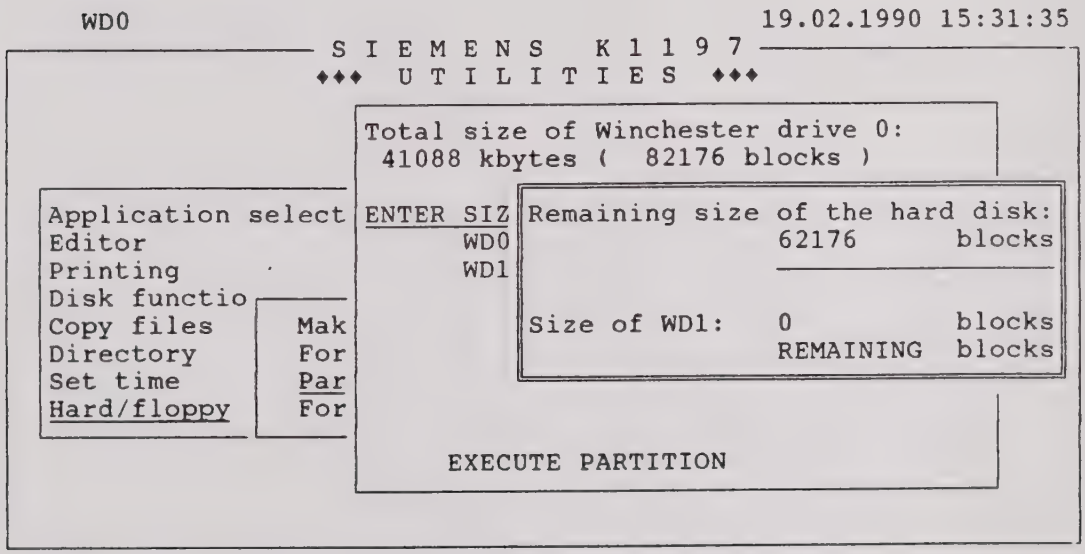


WDx: : PRESS CUROR RIGHT TO SELECT A MENU FOR ENTERING THE SIZE OF WDx.

EXECUTE PARTITION : A WARNING IS GIVEN THAT **ALL FILES WILL BE LOST ON THE HARD DISK.** ACKNOWLEDGEMENT IS REQUIRED BEFORE THE PARTITIONING IS COMPLETED.

HARD/FLOPPY DISK INITIALIZATION

PARTITION HARD DISK MENU

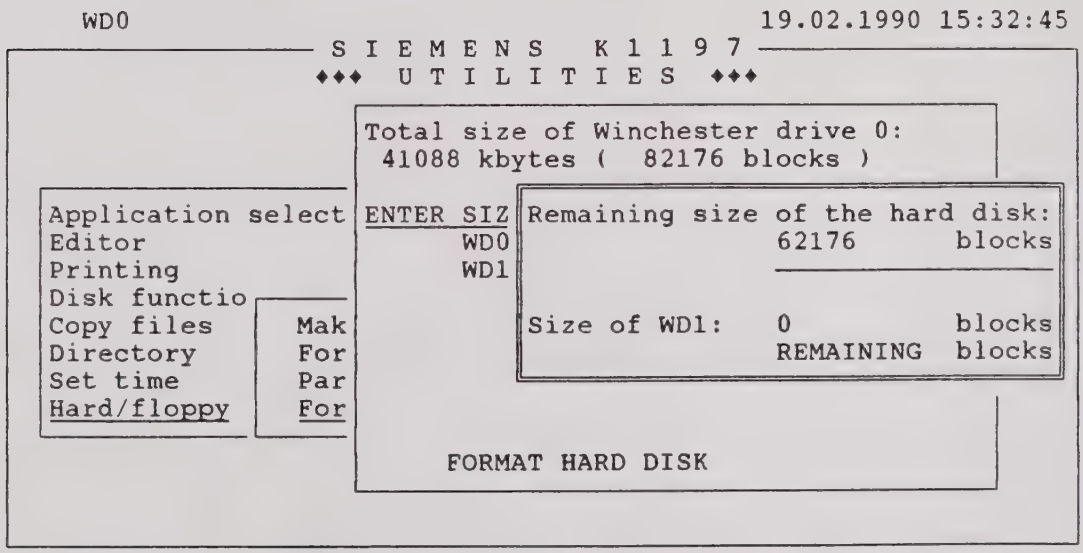


1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

SIZE OF WDx: : TYPE THE DESIRED SIZE OF THE CHOSEN HARD DISK DRIVE IN BLOCKS OR JUST PRESS RETURN ON THE 'REMAINING blocks' WHICH GIVES THIS PARTION ALL REMAINING BLOCKS.

HARD/FLOPPY DISK INITIALIZATION

FORMAT HARD DISK MENU



1QUIT_MENU 2HELP 3COMMAND 4 5 6 7 8

THIS MENU WORKS JUST LIKE **PARTITION HARD DISK MENU** EXCEPT THE TEXT **EXECUTE PARTITION** IS REPLACED BY **FORMAT HARD DISK** AND BEFORE PARTIONING THE HARD DISK WILL BE FORMATTED.

EXERCISE 1

1. FORMATTING

1.1. Reset the K1197.

1.2. Insert a blank disk into drive 0.

1.3. Format the disk in drive 0.

1.4. Create a filesystem on the disk in drive 0.

EXERCISE 1 cont'd

2. DISPLAYING DIRECTORIES

2.1. Insert the exercise disk into drive 0.

2.2. Get a short directory list of all files on drive 0.

2.3. Display a long directory list of all files named "ED-EX-?", where the wildcard '?' stands for a single character.

EXERCISE 1 cont'd

3. COPYING FILES

3.1. Insert the exercise disk into drive 0.

3.2. Make the exercise disk write protected (Software Write Protect).

3.3. Insert the blank (formatted) disk into drive 1.

3.4. Copy all files which filenames start with "ED" from the exercise disk to the blank disk.

EXERCISE 1 cont'd

4. RENAMING AND REMOVING FILES

4.1. Change the name of the file "ED-XXX" on disk drive 1 to "ED-YYY".

4.2. Display the directory of disk drive 1.

4.3. Remove the file "ED-ZZZ" on disk drive 1.

4.4. Display the directory again and check if the file has been removed.

SOLUTION 1

1. FORMATING

1.1. Press the **RESET** button.

1.2. Insert a blank disk into the left hand floppy disk drive.

1.3.

1.3.1 Open the **HARD/FLOPPY DISK INITIALIZATION MENU.**

1.3.2 Move the cursor to the field **'Format diskette'** and press RETURN.

1.3.3 Select **'DRIVE:'** DRO and **'DISK TYPE:'** DOUBLE DENSITY.

1.3.4 Execute **FORMAT FLOPPY DISK.**

1.4.

1.4.1 Move the cursor to the field **'Make filesystem'** and press RETURN.

1.4.2 Enter a volume name and execute **MAKE FILESYSTEM.**

SOLUTION 1 cont'd

2. DISPLAYING DIRECTORIES

2.1. Insert the exercise disk into the lert hand floppy disk drive.

2.2

2.2.1 Open the **DIRECTORY MENU.**

2.2.2 Activate **DRO.** Enter the file name '*' and execute **DIRECTORY SHORT.**

2.3

2.3.1 Go to the **DIRECTORY MENU.**

2.3.2 Activate **DRO.** Enter the file name "ED-EX-?" and execute **DIRECTORY LONG.**

SOLUTION 1 cont'd

3. COPYING FILES

3.1. Insert the exercise disk into the left hand floppy disk drive.

3.2.

3.2.1 Open the **DISK FUNCTIONS MENU.**

3.2.2 Move the cursor to the field **'FILE SYSTEM PROTECTION'** and press RETURN.

3.2.3 Select drive **DRO** and execute **PROTECT FILESYSTEM.**

NOTE: A 3 1/2" floppy disk is also write protected (Hardware Write Protect) if you can look through the hole in the upper right corner.

SOLUTION 1 cont'd

3. COPYING FILES

3.3. Insert the formatted disk into the right hand floppy disk drive.

3.4.

3.4.1 Open the **COPY FILES MENU.**

3.4.2 Select:

COPY FROM:
– DRIVE: DRO
– FILE NAME: ED*

3.4.3 Select:

COPY TO:
– DRIVE: DR1

3.4.4 Execute **COPY FILE(S)**

SOLUTION 1 cont'd

4. RENAMING AND REMOVING FILES

4.1

4.1.1 Open the **DISK FUNCTIONS MENU.**

4.1.2 Move the cursor to the text **'RENAME FILE'** and press RETURN.

4.1.3 Activate **DR1.** Enter the source filename (**OLD NAME:**) **ED-XXX .** Enter the destination filename (**NEW NAME:**) **ED-YYY.**

4.1.4 Execute **RENAME FILE.**

4.2. Execute **DIRECTORY LONG** in the **DIRECTORY MENU** to display the directory of drive 1.

SOLUTION 1 cont'd

4. RENAMING AND REMOVING FILES

4.3

4.3.1 Open the **DISK FUNCTIONS MENU.**

4.3.2 Move the cursor to the field **'REMOVE FILE'** and press RETURN.

4.3.3 Activate **DR1.** Enter the filename **ED-ZZZ.**

4.3.4 Execute **REMOVE FILE.**

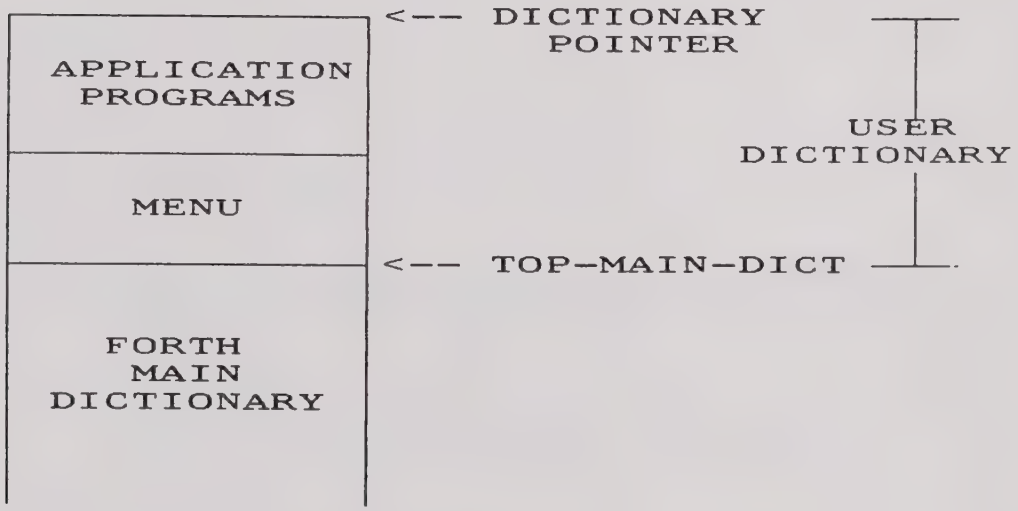
4.4. Execute **DIRECTORY LONG** in the **DIRECTORY MENU** to display the directory of drive 1 again.

PROGRAM USAGE

SUMMARY

- THE STRUCTURE OF THE DICTIONARY
- HOW TO RESET THE DICTIONARY POINTER
- HOW TO CREATE A SOURCE FILE
- HOW TO COMPILE A SOURCE FILE
- HOW TO SAVE THE USER DICTIONARY TO A BINARY FILE
- HOW TO LOAD A BINARY FILE ON TOP OF THE MAIN DICTIONARY

THE STRUCTURE OF THE DICTIONARY



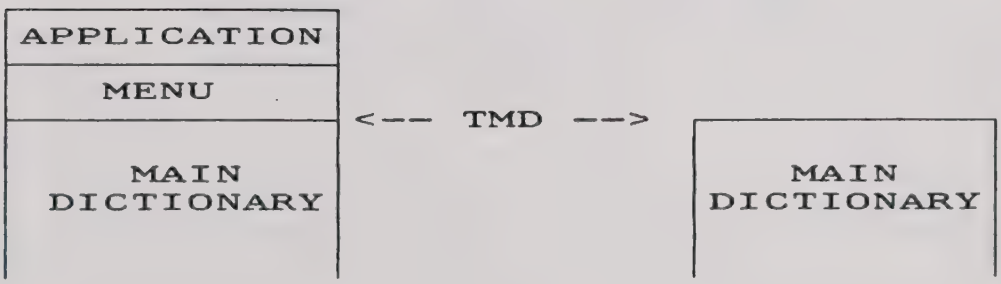
- THE APPLICATION PROGRAMS ARE USER DEFINED PROGRAMS

- THE MENU SYSTEM IS USUALLY CONTAINED IN THE **ALOAD** FILE

HOW TO RESET THE DICTIONARY POINTER

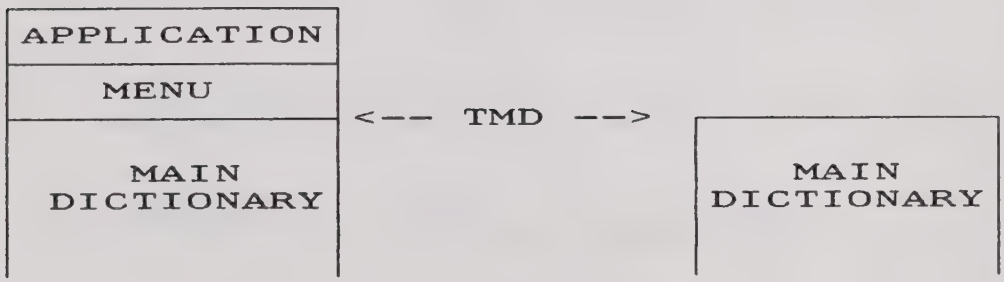
COLD

- 'FORGETS' EVERYTHING ABOVE TOP-MAIN-DICT
- IS THE COLD START PROCEDURE TO ADJUST THE DICTIONARY POINTER TO THE TOP OF THE MAIN DICTIONARY
- MAY BE CALLED FROM THE CONSOLE TO REMOVE APPLICATION PROGRAMS
- USAGE: COLD



HOW TO RESET THE DICTIONARY POINTER

COLD



NOTE: THIS MEANS ANY APPLICATION PROGRAM, INCLUDING THE MENU SYSTEM, IS NOT EXISTENT ANY LONGER AFTER THE COMMAND COLD HAS BEEN ENTERED.

HOW TO CREATE A SOURCE FILE

EDIT

- CALLS THE EDITOR. THE SOURCE FILE WILL BE LOADED INTO THE EDITOR BUFFER IF ALREADY EXISTING.

- IF THE FILE DOESN'T EXIST, A NEW FILE WILL BE CREATED.

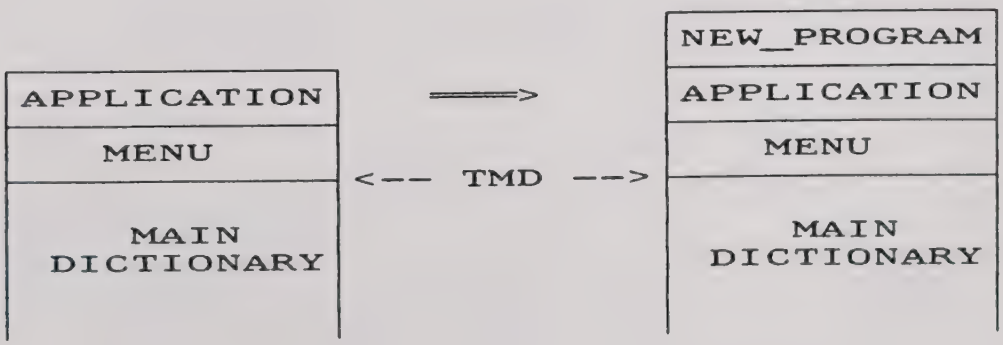
- EXAMPLE: EDIT
  ENTER FILENAME: ...

- AFTER EDITING OF THE SOURCE CODE YOU MAY USE FUNCTION KEY F6 TO SAVE THE SOURCE FILE.

HOW TO COMPILE A SOURCE FILE

EXEC

- COMPILES THE SPECIFIED SOURCE FILE ON TOP OF THE CURRENT DICTIONARY
- THE USER DICTIONARY IS EXTENDED, NOT OVERWRITTEN
- IT IS POSSIBLE TO COMPILE A SOURCE FILE ON TOP OF THE CURRENT DICTIONARY MORE THAN ONCE
- EXAMPLE: " NEW_PROGRAM" EXEC



HOW TO SAVE THE USER DICTIONARY TO A BINARY FILE

SAVEB

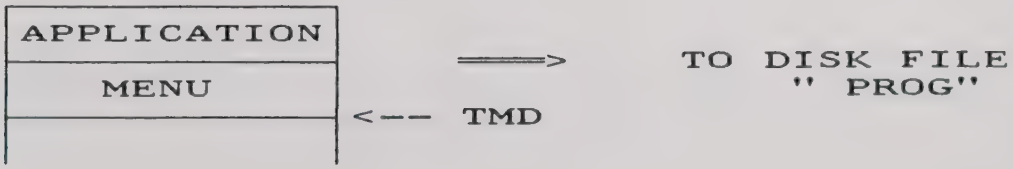
- SAVES THE PRESENT USER DICTIONARY (EVERYTHING ABOVE TOP-MAIN-DICT) INTO THE SPECIFIED BINARY FILE ON DISK

- THE FILE MAY BE LOADED WITH THE EXECB COMMAND

- IF THE FILENAME ALREADY EXISTS ON THE SPECIFIED DISK, IT WILL BE OVERWRITTEN

- THE ENTIRE DICTIONARY IS NOT CHANGED

- EXAMPLE: " PROG" SAVEB

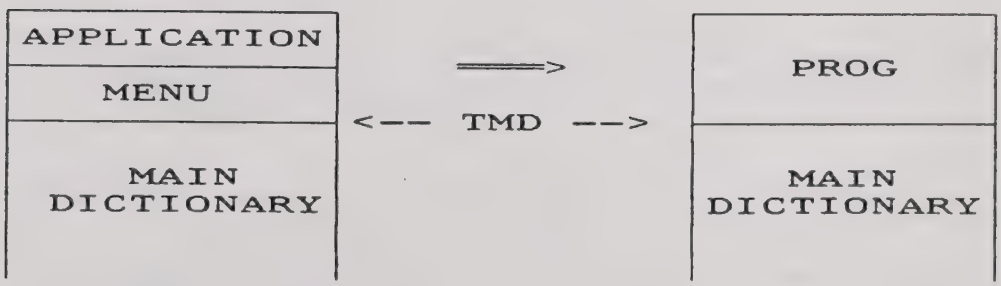


HOW TO LOAD A BINARY FILE ON TOP OF THE MAIN DICTIONARY

EXECB

- LOADS THE SPECIFIED BINARY FILE (SAVED WITH THE SAVEB COMMAND) FROM THE DISK ON TOP OF THE MAIN DICTIONARY

- EXAMPLE: " PROG" EXECB



NOTE: THE PREVIOUS USER DICTIONARY WILL BE OVERWRITTEN.

EXERCISE 1

1. Power up the K1197 (or reset it if power is already up).

2. Give control to the FORTH window.

3. Compile the source file DEMO.SRC from the exercise disk.

4. Run the program named DEMO .

5. Save the binary program to the exercise disk with the name DEMO.BIN.

6. Remove the program DEMO from the dictionary using the command **FORGET DEMO.**

7. Try to run the program DEMO.

8. Load the binary file DEMO.BIN from the exercise disk and run the binary program DEMO again.

SOLUTION 1

1. Press the power switch to power up the K1197 or use the reset button to reset it if power is already up.

2. Press the function key F3 (COMMAND) to open the FORTH window.

3. Select the appropriate drive by typing DR0 for drive 0 or DR1 for drive 1.
   Enter the FORTH command
   " DEMO.SRC" EXEC to load the source file.

4. Run the program by typing the name **DEMO** followed by RETURN.

5. Enter the FORTH command
   " DEMO.BIN" SAVEB to save the binary file.

6. Enter the FORTH command **FORGET DEMO** to remove the program DEMO from the dictionary.

SOLUTION 1 cont'd

7. You can't run the program DEMO, because it is removed from the dictionary.

8. Enter the FORTH command
   " DEMO.BIN" EXECB to load the binary file.
   Run the binary program by typing the name **DEMO** followed by RETURN.

K1197 EDITOR

SUMMARY

- INTRODUCTION
- EDIT SESSION INITIATION
- EDITOR COMPONENTS
- CURSOR KEYS
- EDITING KEYS
- CHARACTER KEYBOARD
- FILE ACCESS AND EXIT
- SEARCH KEYS

K1197 EDITOR

SUMMARY

- CHARACTER SETS KEYS
- ENTRY MODE KEY
- BLOCK COMMANDS
- CHANGING MODES
- COLOR EDITING KEYS

INTRODUCTION

THE K1197-EDITOR IS A SCREEN EDITOR, WHICH MAY BE USED TO CREATE OR CHANGE FILES.
ALL FEATURES ARE ACCESSIBLE BY USE OF FUNCTION KEYS.

EDITOR SCREEN

1	Edit:TEST Row:1 Colum:1 CS:ASCII CHAR:$^{4}_{1}$ 1990-01-01 12:10:00							
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20	CMD_LINE							
21	Cf1	Cf2	Cf3	Cf4	Cf5	Cf6	Cf7	Cf8
22	Sf1	Sf2	Sf3	Sf4	Sf5	Sf6	Sf7	Sf8
23	f1	f2	f3	f4	f5	f6	f7	f8

INTRODUCTION

EDITOR SCREEN

LINE No	NAME	CONTENT/DESCRIPTION
1	STATUS LINE	SHOWS FILENAME, CURSOR POSITION, AND CHARACTER SET
2 - 19	DISPLAY	TEXT DISPLAY
20	COMMAND LINES	USED FOR USER INPUT AND ERROR MESSAGES
21	Cf(1-8)	CONTROLLED FUNCTION KEYS CTRL+(f1-f8)
22	Sf(1-8)	SHIFTED FUNCTION KEYS SHIFT+(f1-f8)
23	f(1-8)	FUNCTION KEYS (f1-f8)

INTRODUCTION

SIZE CONSTRAINTS

- THE EDITOR WORKS IN A 64 KBYTE BUFFER

- THE ENTIRE FILE HAS TO BE IN THAT BUFFER FOR EDITING

- THEREFORE, AN EDITOR FILE MUST NOT EXCEED 64 KBYTES

INTRODUCTION

FEATURES

- INSERT/REPLACE MODE
- ERASE AND INSERT SINGLE LINES
- DUPLICATE AND TRANSFER SINGLE LINES/BLOCKS
- JOIN TWO CONSECUTIVE LINES TOGETHER
- INPUT NEW FILE IN EDITOR BUFFER
- INSERT A FILE
- SELECT DISK DRIVE
- SEARCH FUNCTION (SEARCH FOR CHARACTER STRING)
- REPLACE FUNCTION (REPLACE ONE CHARACTER STRING BY ANOTHER)

INTRODUCTION

FEATURES

- DIFFERENT CHARACTER SETS:
  (ASCII, EBCDIC, HEX, TTX, JIS8)

- DIFFERENT ENTRY MODES:
  (NORMAL, HEX, LITERAL)

- ABILITY TO EDIT FULLY COLORED
  VIDEO-SCREEN
  COLORS: WHITE, GREEN, YELLOW, CYAN,
  RED, BLUE, MAGENTA, BLACK

- FOREGROUND/BACKGROUND COLOR TOGGLE

EDIT SESSION INITIATION

EDIT

- WITH THIS FORTH COMMAND, THE EDITOR IS STARTED **INTERACTIVELY**
- THE SYSTEM WILL ASK FOR A FILENAME

" filename" EDITF

- THE EDITOR CAN ALSO BE STARTED ENTERING THE FILENAME AS A **FORTH-STRING PARAMETER**, FOLLOWED BY THE **FORTH COMMAND EDITF**
- THE SPEZIFIED FILE WILL BE AUTOMATICALLY LOADED

EDITOR MENU

- THE "NORMAL" WAY TO INVOKE THE EDITOR IS TO CALL IT FROM THE EDITOR MENU

EDITIOR COMPONENTS

CHARACTER KEYBOARD

USING THE KEYBOARD THE USER CAN ENTER ANY TEXT AS WITH A NORMAL TYPEWRITER

FUNCTION KEYS

THE FUNCTION KEYS PROVIDE THE MAJORITY OF THE EXTRA FUNCTIONS EXPECTED IN A SCREEN EDITOR

CURSOR KEYS

THERE ARE 4 GROUPS OF CURSOR KEYS THEY ARE USED TO PROVIDE QUICK CURSOR MOVEMENT AND COLOR PAINTING

CURSOR KEYS

NORMAL

- MOVES THE CURSOR IN THE DIRECTION OF THE ARROW
- MOVEMENT CONTINUES UNTIL A BORDER IS REACHED (TOP OR BOTTOM OF FILE)
- AT THE END OF THE LINE (COLUMN 1 OR 80) THE CURSOR MOVES TO ANOTHER (PREVIOUS OR NEXT) LINE

SHIFTED

- PERFORMS THE SAME TASK AS THE UNSHIFTED KEY
- ADDITIONALLY, THE CURRENTLY SELECTED COLOR COMBINATION (WHICH IS DISPLAYED IN THE SET ATTR (SF6) FUNCTION KEY FIELD) IS ALSO APPLIED TO THE CHARACTER WHERE THE CURSOR MOVES ONTO

CURSOR KEYS

CONTROLLED

-->MOVES TO THE FIRST CHARACTER OF THE NEXT WORD

<--MOVES TO THE FIRST CHARACTER OF THE PREVIOUS WORD

↑ MOVES 1 SCREEN (18 LINES) UP

↓ MOVES 1 SCREEN DOWN

SHIFTED + CONTROLLED

-->MOVES TO THE RIGHT OF THE LAST NON-BLANK CHARACTER OF THE LINE

<--MOVES TO THE FIRST CHARACTER OF THE LINE

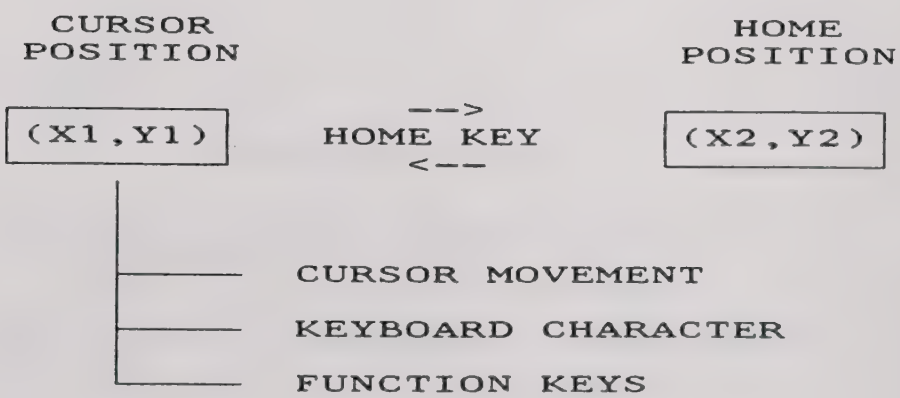
↑ MOVES TO THE TOP OF THE FILE

↓ MOVES TO THE BOTTOM OF THE FILE

CURSOR KEYS

HOME KEY

- MOVES THE CURSOR TO WHERE IT WAS BEFORE THE LAST CURSOR MOVEMENT
- THIS ALLOWS A USER TO RETURN TO THE PREVIOUS POSITION IN CASE OF A CURSOR MOVEMENT ERROR OR IF FLIPPING BETWEEN TWO LOCATIONS IS DESIRED



EDITING KEYS

INS/REP F1

INSERT/REPLACE MODE

- TOGGLES INSERT/REPLACE MODE
- CURRENT STATE IS HIGHLIGHTED IN BLUE BACKGROUND

INS LINE F2

INSERT LINE

- INSERTS A LINE REGARDLESS OF INSERT/REPLACE MODE
- THE LINE IS INSERTED BELOW THE CURSOR

EDITING KEYS

CUT CHAR F3

CUT CHARACTER

- REMOVES THE CHARACTER AT THE CURSOR POSITION AND PUTS IT INTO THE PASTE BUFFER
- MOVES THE TAIL OF THE LINE ONE SPACE TO THE LEFT

CUT LINE F4

CUT LINE

- REMOVES THE CURRENT LINE AND PUTS IT INTO THE PASTE BUFFER
- THE CURSOR IS MOVED TO THE BEGINNING OF THAT LINE WHICH REPLACES THE CUT LINE

EDITING KEYS

PASTE F5

PASTE FROM BUFFER

- PUTS THE LAST CUT ITEM (CHARACTER OR LINE) INTO THE EDITOR BUFFER AT THE CURRENT CURSOR POSITION

- THIS ALLOWS LINES OR CHARACTERS TO BE MOVED AND COPIED

- THE INSERTION OF A CUT LINE IS DONE ABOVE THE CURSOR, WHEREAS A CUT CHARACTER IS ALWAYS INSERTED BEFORE THE CURSOR POSITION

- THIS ALLOWS EASY RECOVERY FROM ERRONEOUS USE OF CUT KEYS

EDITING KEYS

JOIN LINE SF3

JOIN TWO CONSECUTIVE LINES TOGETHER

- JOINS TWO CONSECUTIVE LINES TOGETHER IF THE SECOND LINE FITS IN THE SPACE AT THE END OF THE FIRST LINE

- A SPACE IS USED TO SEPARATE THE CONTENTS OF THE TWO LINES AND THE CURSOR IS PLACED AT THE FIRST NON-BLANK CHARACTER OF THE SECOND LINE

CHARACTER KEYBOARD

THE ACTION OF THE CHARACTER KEYBOARD, INCLUDING THE TAB AND BACKSPACE (BS) KEYS DEPENDS ON THE INSERT/REPLACE MODE.

INSERT MODE

- NORMAL CHARACTER KEY

  THE APPROPRIATE CHARACTER IS INSERTED.
  SUBSEQUENT TEXT IS SHIFTED TO THE RIGHT

- TAB KEY

  MOVES THE CURSOR TO THE NEXT TAB POSITION IN THE CURRENT LINE.
  SUBSEQUENT TEXT ON THAT LINE IS MOVED OVER TO THE RIGHT.
  TAB POSITIONS ARE PREDEFINED AT FIXED POINTS (COLUMNS 5, 9, 13 ETC.)

- BACKSPACE

  REMOVES THE CHARACTER BEFORE THE CURSOR AND PUTS IT INTO THE PASTE BUFFER.
  SUBSEQUENT TEXT IS MOVED ONE SPACE TO THE LEFT

CHARACTER KEYBOARD

REPLACE MODE

- NORMAL CHARACTER KEY

  THE APPROPRIATE CHARACTER OVER-WRITES THE CHARACTER AT THE CURSOR POSITION

- TAB KEY

  WORKS LIKE THE TAB KEY IN INSERT MODE BUT SUBSEQUENT TEXT (UP TO 4 CHARACTERS) ON THE CURRENT LINE IS REPLACED BY SPACES

- BACKSPACE

  REPLACES THE CHARACTER ON THE LEFT SIDE OF THE CURSOR POSITION BY A SPACE AND MOVES THE CURSOR TO THE LEFT.
  THE CONTENTS OF THE PASTE BUFFER IS NOT CHANGED.

FILE ACCESS AND EXIT

SAVE F6

SAVE TO DISK-FILE

- WRITES THE EDITED TEXT TO A FILE WHOSE NAME IS ENTERED BY THE USER
- THE FILE IS REMOVED IF IT EXISTS, CREATED AND WRITTEN
- THE USER IS PROMPTED WITH THE NAME OF THE FILE USED IN THE PREVIOUS FILE ACCESS COMMAND. IT CAN BE REMOVED BY USING CUT LINE OR EDITED USING THE KEYBOARD KEYS.

QUIT F8

QUIT THE EDITOR

- QUITS THE EDITOR
- REQUESTS CONFIRMATION AND INDICATES IF THERE ARE ANY CHANGES SINCE THE LAST SAVE

FILE ACCESS AND EXIT

DR[0,1] SF2

SELECT DRIVE 0 OR 1

- CHANGES THE DEFAULT DRIVE
- CAUSES THE FILE I/O TO BE PERFORMED ON THE SPECIFIED DRIVE

NEW FILE SF4

INPUT NEW FILE

- CAUSES A NEW FILE TO BE READ FROM THE DEFAULT DRIVE
- PROMPTS THE CURRENT FILE NAME, WHICH CAN BE MODIFIED
- AFTER PRESSING RETURN THE NEW FILE IS READ
- **CAUTION:** THE FILE IN THE EDITOR BUFFER IS OVERWRITTEN. IT SHOULD BE PREVIOUSLY SAVED.

FILE ACCESS AND EXIT

INSERT FILE CF7

INSERT FILE

- INSERT THE TEXT OF A FILE WHOSE NAME IS ENTERED BY THE USER

- THE TEXT IS INSERTED BELOW THE CURSOR

SEARCH KEYS

SRCH/REP F7

SEARCH FOR STRING

- PROMPTS FOR ENTRY OF A STRING TERMINATED BY A RETURN
- HEX AND LITERAL ENTRIES ARE ALLOWED
- THE STRING IS THEN MAPPED WITH THE TEXT IN THE FILE ACCORDING TO THE CURRENT CHARACTER SET SELECTED
- IF THE STRING IS FOUND, THE CURSOR IS MOVED TO THE START OF THE STRING, OTHERWISE AN ERROR MESSAGE IS DISPLAYED
- THE SEARCH WILL WRAP AROUND THE BOTTOM OF THE FILE UNTIL THE CURRENT CURSOR POSITION
- ONE CAN USE THE WILDCARD '?' FOR ANY CHARACTER IN THE SEARCH STRING

SEARCH KEYS

SRCH/REP F7

SEARCH FOR STRING AND REPLACE

- PROMPTS FOR ENTRY OF A STRING WHICH SHOULD BE REPLACED
- AFTER PRESSING "-->" THE USER IS PROMPTED FOR ENTRY OF A REPLACE STRING
- NOW THE USER CAN DECIDE, WHETHER HE WANTS TO REPLACE ONLY ONE OR ALL OCCURENCES OF THIS STRING IN THE TEXT:
  - SF2 REPLACE ONLY ONE STRING
  - SF3 REPLACE ALL OCCURENCES OF THE STRING

SEARCH KEYS

<next> SF1

SEARCH FOR NEXT OCCURENCE OF STRING

- MOVES THE CURSOR TO THE FIRST CHARACTER OF THE NEXT OCCURENCE OF THE SEARCH STRING WITHOUT RE-ENTERING THAT STRING

CHARACTER SETS KEY

CS:[ASCII,EBCDIC,HEX,TTX,JIS8]

CF8 SELECT CHARACTER SET

- ALLOWS ONE OF THE 5 CHARACTER SETS, NAMELY ASCII, EBCDIC, HEX, TTX (TELETEX) OR JIS8 (KATAKANA) TO BE CYCLICALLY SELECTED; JIS8 IS OPTIONAL

- ALL CHARACTERS ENTERED HAVE THE CURRENT CHARACTER SET INDICATION AS PART OF THEIR ATTRIBUTES

ENTRY MODE KEY

[NORMAL,HEX,LITERAL] SF8

SELECT ENTRY MODE

ALLOWS ONE OF THE 3 ENTRY MODES, NAMELY NORMAL, HEX, OR LITERAL MODE TO BE CYCLICALLY SELECTED

NORMAL MODE

- ALLOWS ALL CHARACTERS TO BE ENTERED EXCEPT BS, TAB, CR, AND CONTROL-P (TELETEX CONTROL-P)

ENTRY MODE KEY

HEX MODE

- ALL CHARACTERS ARE ENTERED A NIBBLE AT A TIME
- ONLY HEX CHARACTERS ARE ALLOWED FOR ENTRY (0-9,a-f,A-F)
- THE FIRST HEX DIGIT IS DISPLAYED BESIDE THE HEX PROMPT AT THE LOWER RIGHT OF THE SCREEN
- AFTER A SECOND NIBBLE HAS BEEN ENTERED THE HEX CHARACTER APPEARS AT THE CURSOR POSITION

LITERAL MODE

- ALLOWS THE ENTRY OF ALL CHARACTERS. THE CHARACTERS BS, TAB, AND CR ARE TREATED AS NORMAL CHARACTERS AND THEREFORE ARE NOT DIRECTLY EXECUTED.

BLOCK COMMANDS

MARK CF1

MARK BLOCK

- THE LINE WHERE THE CURSOR IS PLACED GETS MARKED

- TO HANDLE BLOCKS OF MORE THEN ONE LINE THE USER HAS TO SET TWO MARKS, ONE AT THE BEGINNING AND THE OTHER ONE AT THE END OF THE BLOCK

UNMARK CF2

UNMARK

- USED FOR REMOVING THE MARK

BLOCK COMMANDS

BLK MOVE — CF3

BLOCK MOVE

- BEFORE MOVING, THE BLOCK MUST BE MARKED (SEE MARK)
- THE BLOCK IS MOVED BELOW THE CURSOR
- THE DESTINATION MUST NOT BE INSIDE THE MARKED BLOCK

BLOCK COMMANDS

BLK COPY CF4

BLOCK COPY

- BEFORE COPYING, THE BLOCK MUST BE MARKED (SEE MARK)
- THE BLOCK IS COPIED BELOW THE CURSOR
- THE DESTINATION MUST NOT BE INSIDE THE MARKED BLOCK

BLOCK COMMANDS

BLK DEL CF5

BLOCK DELETE

- BEFORE DELETING, THE BLOCK MUST BE MARKED (SEE MARK)
- REMOVES THE BLOCK AND PUTS IT INTO THE PASTE BUFFER

BLK SAVE CF6

BLOCK SAVE

- BEFORE SAVING, THE BLOCK MUST BE MARKED (SEE MARK)
- SAVES THE BLOCK TO A FILE WHOSE NAME IS ENTERED BY THE USER

CHANGING MODE

MOD:MISC\COLOR SF7

MISCELLANEOUS\COLOR

- CHANGES THE MEANING OF THE CTRL FUNCTION KEYS

- THE STANDARD SETTING IS "MISCELLANEOUS" (E.G. MARK (CF1), UN-MARK (CF2)...).
  THE FUNCTIONS ARE DESCRIBED ABOVE

- THE FUNCTIONS FOR THE COLOR MODE ARE DESCRIBED BELOW

COLOR EDITING KEYS

CTRL FUNCTION KEYS (IN COLOR MODE)

- ALLOWS THE SELECTION OF A FOREGROUND/BACKGROUND COLOR COMBINATION
- THE DEFAULT COLOR IS WHITE CHARACTERS ON A BLACK BACKGROUND
- FOREGROUND COLORS: WHITE, GREEN, YELLOW, CYAN
- BACKGROUND COLORS: RED, BLUE, MAGENTA, BLACK
- THE CURRENT FOREGROUND/BACKGROUND COMBINATION IS REFLECTED BY THE COLORS OF THE SET ATTR (SF6) FUNCTION KEY FIELD

COLOR EDITING KEYS

REVERSE SF5

REVERSE COLORS

- REVERSES THE BACKGROUND AND FOREGROUND COLORS
- THE CURRENT MODE IS REFLECTED BY THE REVERSE AND THE SET ATTR (SF6) FUNCTION KEY FIELDS

[SET ATTR , END ATTR] SF6

SET/END ATTRIBUTE

- AFTER HAVING CHOSEN THE DESIRED COLOR COMBINATION PRESS THE SET ATTR KEY TO SELECT IT
- TEXT WILL BE ENTERED IN THIS COLOR COMBINATION UNTIL THE END ATTR KEY IS PRESSED, WHICH MAKES THE EDITOR RETURN TO THE DEFAULT VALUE (WHITE ON BLACK)

COLOR EDITING KEYS

CHANGING THE COLOR OF PREVIOUSLY ENTERED TEXT:

- POSITION THE CURSOR AT START OF TEXT TO BE PAINTED
- SELECT THE DESIRED COLOR COMBINATION USING THE CTRL FUNCTION KEYS
- USE THE SHIFTED CURSOR KEYS TO PAINT THE TEXT IN THE CURRENT COLOR COMBINATION INDICATED BY THE SET ATTR (SF6) KEY

EXERCISE 1

1. Insert the exercise disk in drive 1. Open the EDITOR MENU and select drive 1.

2. Load the file "ED-EX-TEXT1" on disk drive 1 into the Editor Buffer.

3. Perform the following changes to the file:

   3.1. Remove the entire paragraph painted in blue background.

   3.2. Move the whole paragraph which contains the word "balderdash" to the end of the file.

   3.3. Move the paragraph painted in green foreground three TAB positions to the right.

EXERCISE 1 cont'd

3.4. Replace all occurrences of the dummy number "value" by the correct number "thousand".

3.5. Search for the third occurence of the string "stars" from top of the file and correct it.

4. Save the edited file "ED-EX-TEXT1" under the new filename "ED-EX-MYTEXT1" to disk.

5. Quit the editor session.

EXERCISE 2

1. Insert the exercise disk in drive 1. Open the EDITOR MENU and select drive 1.

2. Edit the new file "ED-EX-MYTEXT2".

3. Carry out the successive jobs:

   3.1. Insert the following characters:

   B C d f $^{B}{}_{S}$ 1 2 ! " $^{L}{}_{F}$ % & /

   ^ @ $^{M}{}_{T}$] # : } ? _ ~ \ ¦ $^{C}{}_{R}$

EXERCISE 2 cont'd

3.2. Enter the following ASCII character string in HEX mode:

```
21 21 21 20 47 4F 4F 44 20 4C
55 43 4B 20 5G 46 4F 52 20 59
4F 55 H2 20 41 4C 4C 20 21 21
21
```

What is wrong inside the given string ?

4. Save the edited file.

5. Quit the editor session.

EXERCISE 3

1. Insert the exercise disk in drive 1. Open the EDITOR MENU and select drive 1.

2. Load the file named "ED-EX-TEXT3" on disk drive 1 into the Editor Buffer.

3. Solve the subsequent tasks:

   3.1. Paint the first paragraph in yellow foreground.

   3.2. Reverse the background and foreground colors of the last two paragraphs.

EXERCISE 3 cont'd

3.3. Paint the whole third paragraph in blue background.

3.4. As the seventh paragraph insert the sentence "The businessman raised his head.". Before writing it select green foreground and red background colors.

4. Save the edited file "ED-EX-TEXT3" under the new filename "ED-EX-MYTEXT3" to disk.

5. Quit the editor session.

SOLUTION 1

1. Insert the exercise disk into the right hand floppy disk drive. Bring up the EDITOR MENU. Select DR1 if not already selected.

2. Enter the filename "ED-EX-TEXT1", and execute EDIT FILE.

3. Perform the following changes to the file:

   3.1. Find the paragraph painted in blue background. Move the cursor to the first line of this paragraph. Hit the function key **CF1 (MARK)** to mark this line. Mark also the last line and press CF5 (**BLK DEL**) to delete the paragraph.

SOLUTION 1 cont'd

3.2. Enter the search mode (press **F7**). Enter the search string "**balderdash**" to find the wanted paragraph. Move the cursor to the first line of this paragraph. Press **CF1 (MARK)** to mark this line. Then move the cursor to the last line of this paragraph and hit **CF1 (MARK)** again. Move the cursor to the end of the file (**SHIFT+ CTRL + ↓**). Press **CF3 (BLK MOVE)** to move the marked paragraph to the end of the file. Then unmark the moved block (**CF2**).

3.3. Make sure that the editor is in insert mode (F1). Move the cursor to the beginning of the first line of the paragraph painted in green foreground. Press the **TAB** key three times. Move the cursor to column 0 of the next line
(**↓** , **SHIFT + CTRL + <--**)
etc.

SOLUTION 1 cont'd

3.4. Enter the search mode (press **F7**) and the search string "**value**". Press "**-->**" and enter the replace string "**thousand**". Hit **SF3** (replace all) to replace all occurrences of the dummy number.

3.5. Move the cursor to the top of the file (**SHIFT + CTRL + ↑**). Search for the string "**stars**". If found press the **<next>** (**SF1**) key twice to find the third occurence of the wanted string. Correct it.

4. Hit the function key **F6** (**SAVE**) and enter the new filename to save the edited file.

5. Press **F8** (**QUIT**) and confirm with "**Y**".

SOLUTION 2

1. Insert the exercise disk into the right hand floppy disk drive. Bring up the EDITOR MENU. Select DR1 if not already selected.

2. Enter the filename "**ED-EX-MYTEXT2**", and execute EDIT FILE.

3. Carry out the successive jobs:

   3.1. Use the **CAPS LOCK** and the **SHIFT** key to enter most of the given characters. The characters

   $^{B}_{S}$ **BACKSPACE**
   $^{L}_{F}$ **LINE FEED**
   $^{C}_{R}$ **CARRIAGE RETURN**
   $^{H}_{T}$ **TAB**

   must be entered in **LITERAL** mode (press the function key **SF8**).

SOLUTION 2 cont'd

3.2. Select **HEX** entry mode (press **SF8**). Insert the character string.

In HEX mode only HEX characters are allowed for entry. G and H are "forbidden" characters.

4. Hit the function key **F6 (SAVE)** to save the edited file.

5. Press **F8 (QUIT)** and confirm with "Y".

SOLUTION 3

1. Insert the exercise disk in drive 1. Bring up the EDITOR MENU. If necessary select DR1 for drive 1.

2. Enter the filename "ED-EX-TEXT3", and execute EDIT FILE.

3. Solve the subsequent tasks:

   3.1. Move the cursor to the beginning of the first paragraph. Change from misc-mode to color mode (Press SF7). Hit the function key CF3 to choose the yellow foreground. Use the SHIFTED cursor keys to move the cursor over the text to be colored.

SOLUTION 3 cont'd

3.2. Move the cursor to the first line, column 0, of the second last paragraph. Choose the foreground and background colors of this paragraph (**CONTROL** function keys). Press the **REVERSE** (**SF5**) key. Reverse the color combination by using the **SHIFTED** cursor keys as a paintbrush. Do the corresponding actions for the last paragraph.

3.3. Move the cursor to the beginning of the third paragraph. Hit the function keys **CF1** and **CF6** to choose the white foreground and the blue background. Use the **SHIFTED** cursor keys to move the cursor over the text to be colored.

SOLUTION 3 cont'd

3.4. Position the cursor at the blank line between the sixth and seventh paragraph. Press the **RETURN** key. Choose the green foreground (**CF2**) and red background (**CF5**) colors. Select this color combination by pressing the **SET ATTR** (**SF6**) key. Insert the given sentence. After that, return to the default color combination by pressing **END ATTR** (**SF6**).

4. Hit the function key **F6** (**SAVE**) and enter the new filename to save the edited file.

5. Press **F8** (**QUIT**) and confirm with "**Y**".

FORTH INTRODUCTION

SUMMARY

- WHAT IS FORTH ?
- WHY CHOOSE FORTH ?
- K1197 IMPLEMENTATION OF FORTH
- FORTH ENHANCEMENTS

WHAT IS FORTH ?

A PROGRAMMING LANGUAGE

- HIGH LEVEL
- LOW LEVEL

AN OPERATING SYSTEM

- INTERPRETER
- COMPILER
- ASSEMBLER
- TEXT-EDITOR
- MEMORY MANAGEMENT
- I/O MANAGEMENT

WHAT IS FORTH ?

A PHILOSOPHY

- FUNCTIONAL STRUCTURES
- MODULAR
- INFORMATION HIDING
- PROTOTYPING

WHY CHOOSE FORTH ?

- INTERACTIVE
- EXTENSIBLE
- STRUCTURED
- FAST
- COMPACT

K1197 IMPLEMENTATION OF FORTH

CONTAINS ELEMENTS OF

- FORTH-79
- FIG-FORTH
- 68K-FORTH

EXTENTIONS AND ENHANCEMENTS

DISK FILE SYSTEM:

HIGH SPEED FILE SYSTEM TO HANDLE VARIABLE LENGTH NAMED FILES

EDITOR:

COLOR EDITOR

K1197 IMPLEMENTATION OF FORTH

EXTENTIONS AND ENHANCEMENTS

DISPLAY:

COLOR SCREEN
MULTIPLE CHARACTER SETS
WINDOW OPERATIONS
ENTRY OF UNPRINTABLE CHARACTERS

TERMINAL INPUT:

FUNCTION KEY CAPABILITY
REMOTE TERMINALS

DATA COMMUNICATION:

TRANSMISSION AND RECEPTION OF SERIAL DATA
TIMER MANIPULATION
SEPERATE CONFIGURATION OF EACH TEST PORT

REAL TIME DISK RECORDING:

DATA STREAM MERGER DSM

CASE CONSTRUCT:

SELECT ONE OF A SET OF POSSIBLE CASES

FORTH ENHANCEMENTS

- EXTENSIONS FOR
  CONTROL STRUCTURES AND VECTORED
  EXECUTION

- EXTENSIONS FOR
  COMFORTABLE USER INTERFACE
  (MENU SYSTEM, FUNCTION KEYS)

- EXTENSIONS FOR
  PROTOCOL DECODING AND TESTING

FORTH CONCEPTS

SUMMARY

- WORD
- DICTIONARY
- INTERPRETER
- RPN – REVERSE POLISH NOTATION
- STACK
- COMPARATIVE OPERATORS AND BOOLEAN FLAGS

WORD

UNIT OF EXECUTION (PROCEDURE, FUNCTION, SUBROUTINE)

- EVERY ELEMENT OF THE LANGUAGE FORTH IS A **WORD** (EXCEPT LITERALS)
- EXECUTABLE PROCEDURE EXECUTED BY SIMPLY CALLING ITS NAME
- MAY BE DEFINED AS A GROUP OF ALREADY EXISTING WORDS
- DEFINITION OF A **WORD** LEAVES A **DICTIONARY** ENTRY
- INFORMATION IS USUALLY PASSED TO AND FROM A **WORD** ON THE **PARAMETER STACK**

DICTIONARY

CONTAINS DEFINITIONS OF WORDS

- THREADED LIST OF **WORDS**
- ONCE A **WORD** HAS BEEN ENTERED IN THE **DICTIONARY** (DEFINED), IT CAN BE EXECUTED INTERPRETIVELY
- **DICTIONARY** ENTRIES ARE DONE BY DEFINING **WORDS**
- **WORDS** ARE THREADED IN THE ORDER THEY ARE DEFINED, NOT IN LEXICAL ORDER
- THE FIRST PART OF THE DICTIONARY IS THE SYTEM DICTIONARY (LIKE A LIBRARY OF USEFUL COMMANDS)

INTERPRETER

IN FORTH THERE ARE 2 TYPES OF INTER-
PRETERS TO BE DISTINGUISHED:

OUTER INTERPRETER

- IDENTIFIES A **WORD** IN THE DICTIONARY USING THE HEADER OF THE WORD

INNER INTERPRETER

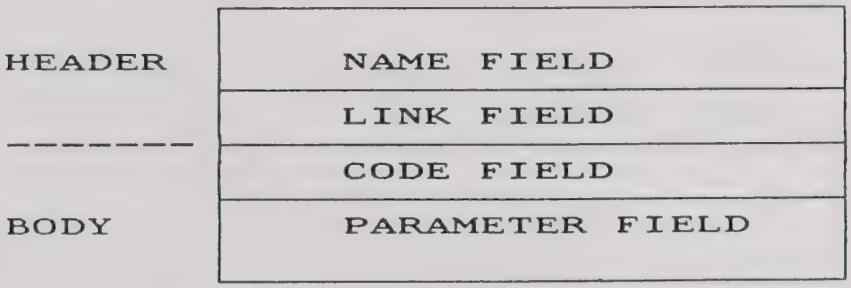
- EXECUTES THE **WORD** USING ITS BODY

INTERPRETER

OUTER INTERPRETER

REMEMBER EACH **FORTH WORD** HAS A DICTIONARY ENTRY.

THE STRUCTURE IS THE FOLLOWING:



THE OUTER INTERPRETER IS A LOOP WHICH COMPARES STRINGS IN THE INPUT STREAM WITH THE NAMES OF FORTH WORDS IN THE DICTIONARY.

THE LATEST DICTIONARY ENTRY IS COMPARED FIRST AND THAN THE PREVIOUS ONE USING THE LINK POINTER FIELD AND SO ON.

INTERPRETER

OUTER INTERPRETER

IF A MATCH IS FOUND, CONTROL IS PASSED TO THE INNER INTERPRETER.

OTHERWISE, THE STRING IS TRIED TO BE INTERPRETED AS A NUMBER, WHICH IS PLACED ON TOP OF STACK (TOS).

IF THIS IS NOT POSSIBLE, SINCE ANY CHARACTER IS NOT A VALID DIGIT ACCORDING TO THE CURRENT **BASE**, THE STRING IS DISPLAYED AT THE SCREEN FOLLOWED BY ??.

INTERPRETER

INNER INTERPRETER

THE CODE POINTER IN THE BODY OF A WORD POINTS TO THE CODE TO BE EXECUTED USING THE PARAMETERS IN THE PARAMETER FIELD.

THE PARAMETERS MAY BE POINTERS TO OTHER WORDS BY WHICH THE CURRENT WORD IS DEFINED. TO CONTROL THIS NESTED EXECUTION IS THE JOB OF THE INNER INTERPRETER.

RPN – REVERSE POLISH NOTATION (POSTFIX NOTATION)

PHILOSOPHY

- THE PARAMETERS ARE NOTED FIRST FOLLOWED BY THE OPERATOR(S)
- THE INPUT IS UNAMBIGUOUS WITHOUT BRACKETS

EXAMPLES:

INFIX		RPN
2 + 4	=	2 4 +
2 / (4+3)	=	2 4 3 + /
2 / 4 + 3	=	2 4 / 3 +

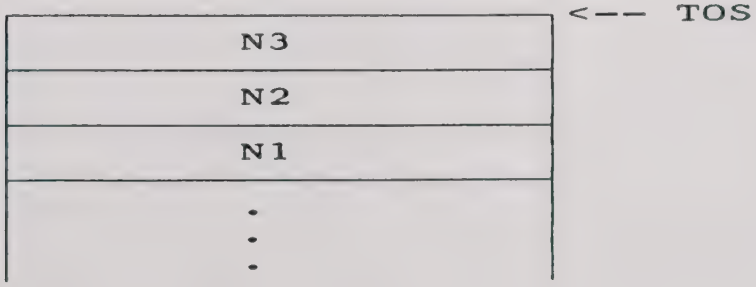
THERE IS A **FORTH WORD** EXISTENT FOR EACH ARITHMETICAL OPERATOR.

RPN IN FORTH MEANS THAT ALL PARAMETERS OF A **FORTH WORD** HAVE TO BE ON THE PARAMETER STACK BEFORE THE WORD IS EXECUTED.

STACK

PARAMETER STACK

- CENTRAL INSTRUMENT FOR PARAMETER PROCESSING
- MAKES MANY AUXILIARY VARIABLES SUPERFLUOUS
- ORGANIZED AS **LIFO** (LAST IN – FIRST OUT) BUFFER
- EACH ENTRY CONTAINS 4 BYTES



- EACH ENTRY OR EXTRACTION OF THE TOS – ELEMENT CHANGES A STACK-POINTER

STACK

RETURN STACK

- **LIFO** ORGANIZED (LIKE PARAMETER STACK)
- USED FOR:
  - LOOP OPERATIONS
  - SAVING OF RETURN ADDRESSES IN NESTED OPERATIONS
  - TEMPORARY STORING OF PARAMETERS (CAUTION !!)

STACK

STACK NOTATION OF FORTH WORDS

- GENERAL FORM:

  (BEFORE EXECUTION -- AFTER EXEC.)

- FOR MULTIPLE STACK ENTRIES "\" IS USED TO SEPERATE THE ENTRIES

- THE LEFT-HAND ENTRY IS THE LOWEST ITEM ON THE STACK, THE RIGHT-HAND THE HIGHEST

EXAMPLES:

(V1\V2 -- V3)

READ AS:

EXPECTS V1 UNDER V2, LEAVES V3

STACK

STACK NOTATION OF FORTH WORDS

EXAMPLES:

* (N1\N2 -- P)

THE WORD * EXPECTS TWO VALUES ON THE STACK, THE TWO NUMBERS TO BE MULTIPLIED AND LEAVES THE RESULT OF THIS MULTIPLICATION.

USAGE: 3 4 * . CR

12 OK

STACK DESCRIPTION:

(MULTIPLICAND 1 \ MULTIPLICAND 2 -- PRODUCT)

STACK

PARAMETER STACK OPERATIONS

- INPUT
- OUTPUT
- DUP
- SWAP
- ROT
- OVER
- PICK
- DROP

STACK

STACK INPUT

THE PARAMETER STACK IS FILLED BY

- KEYBOARD ENTRIES OF NUMBERS OR LITERALS
  (ONLY DIGITS ACCORDING TO THE CURRENT **BASE** ARE ALLOWED)

- CALLING **FORTH WORDS** WHICH LEAVE ENTRIES
  (THE NUMBER OF ENTRIES A **WORD** LEAVES IS NOT FIXED)

STACK

STACK OUTPUT

THE PARAMETER STACK IS EMPTIED BY CALLING FORTH WORDS WHICH EXTRACT STACK ENTRIES

.	REMOVES TOS ENTRY AND OUTPUTS IT AS VALUE ACCORDING TO THE CURRENT BASE
S.	OUTPUTS TOS ENTRY WITHOUT MODIFYING THE STACK
.S	OUTPUTS TOP 4 STACK ENTRIES WITHOUT MODIFYING THE STACK
STACK	OUTPUTS COMPLETE STACK WITHOUT MODIFYING IT

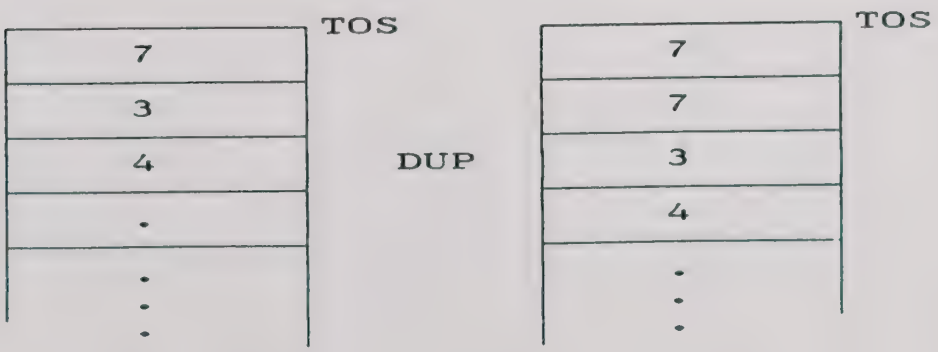
STACK

DUP (N1 -- N1\N1)

DOUBLES TOS ENTRY

EXAMPLE:

```
4 3 7        CR      OK

DUP          CR      OK

. . . .      CR      7 7 3 4 OK
```



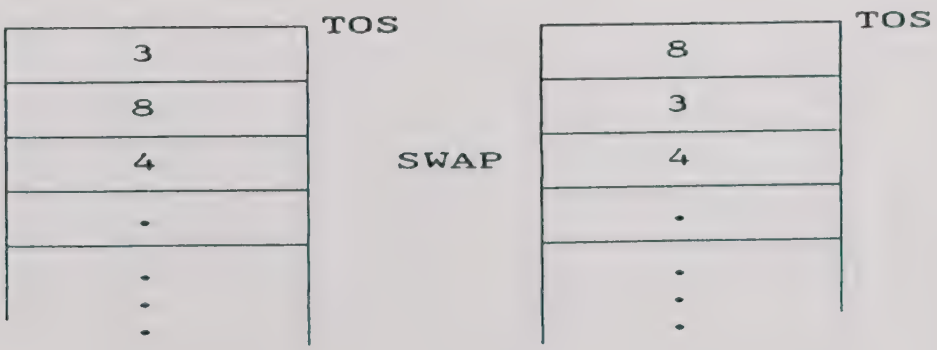
STACK

SWAP (N1\N2 -- N2\N1)

SWAPS THE TWO TOP ENTRIES

EXAMPLE:

```
4 8 3        CR      OK

SWAP         CR      OK

. . .        CR      8 3 4 OK
```



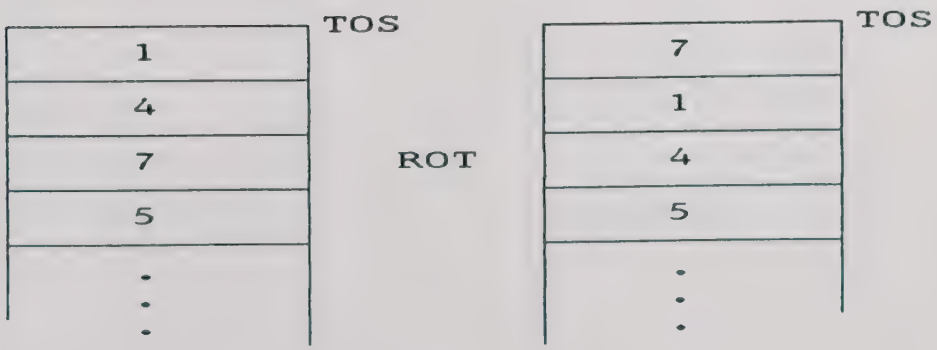
STACK

ROT (N1\N2\N3 -- N2\N3\N1)

ROTATES 3 ENTRIES SUCH THAT THE THIRD ENTRY LIES ON TOP

EXAMPLE:

```
5 7 4 1     CR      OK

ROT         CR      OK

. . . .     CR      7 1 4 5 OK
```



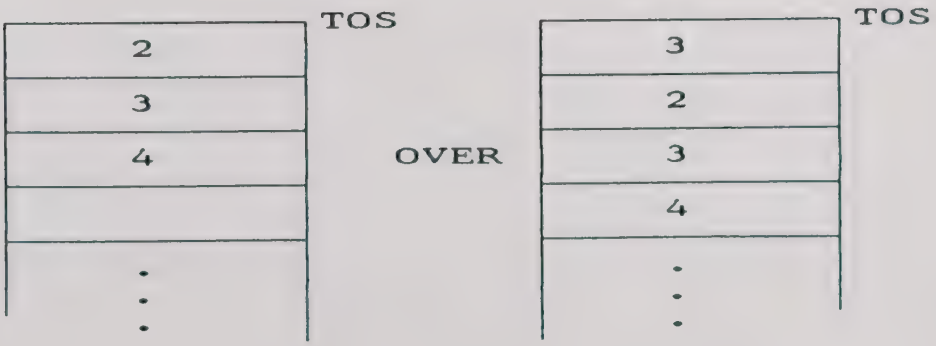
STACK

OVER (N1\N2 -- N1\N2\N1)

COPIES THE VALUE BELOW TOS ON TOS

EXAMPLE:

```
4 3 2        CR      OK

OVER         CR      OK

. . . .      CR      3 2 3 4 OK
```



STACK

PICK (N -- M)

LEAVES A COPY (M) OF THE Nth STACK ENTRY ON TOS

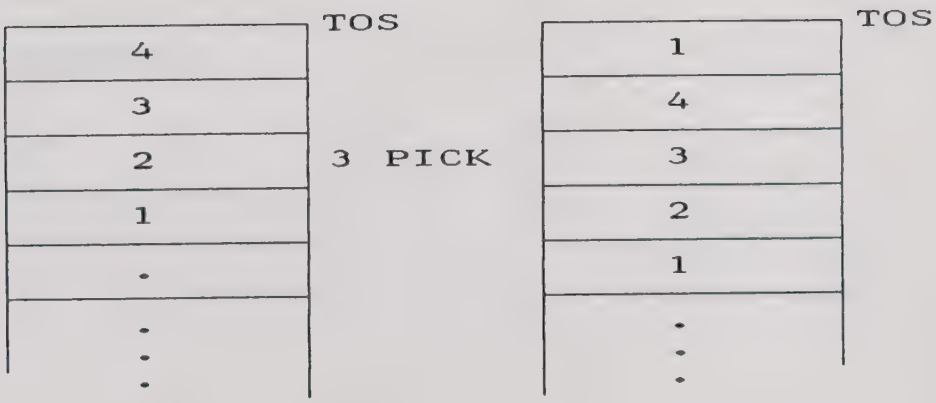
N STARTS AT 0, THUS 0 PICK IS THE SAME AS DUP. 1 PICK IS THE SAME AS OVER.

EXAMPLE:

```
1 2 3 4    CR      OK

3 PICK     CR      OK

. . . . .  CR      1 4 3 2 1 OK
```

STACK

PICK (N -- M)



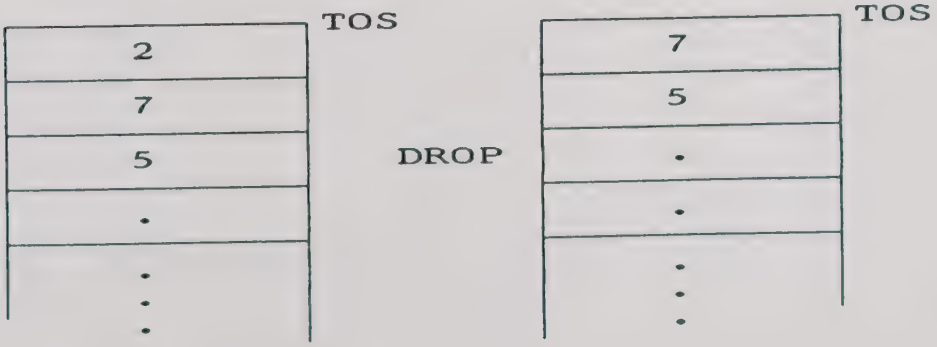
STACK

DROP (N --)

DROPS TOS ENTRY

EXAMPLE:

```
5 7 2        CR      OK

DROP         CR      OK

. . .        CR      7 5 STACK EMPTY OK
```



COMPARATIVE OPERATORS AND BOOLEAN FLAGS

THE OPERATOR " < " (LESS THAN) MAY SERVE AS AN EXAMPLE

```
( D1\D2 -- BOOLEAN FLAG )
```

A BOOLEAN FLAG REPRESENTS ONE OF THE LOGICAL VALUES "TRUE" OR "FALSE"

THE REPRESENTATION OF "TRUE" IS 1
(OR ANY NONZERO NUMBER)
THE REPRESENTATION OF "FALSE" IS 0

EXAMPLE:

```
1 2 < .    CR    1 OK
2 1 < .    CR    0 OK
```

EXERCISE 1

CONVERT THE FOLLOWING INFIX-NOTATIONS TO POSTFIX NOTATIONS:

a) (4 - 3) * (6 + 2)

b) 5 - 7 + 3 * (8 - 3)

c) (4 + 2 * (6 + 12))/((7 - 4) * (2 + 3))

d) c (a + b)

e) (3a - b) / 4 + c

EXERCISE 2

CONVERT THE FOLLOWING POSTFIX NOTATIONS TO INFIX NOTATIONS

a) 2 8 * 4 2 + /

b) 1 2 - 3 * 7 + 2 /

c) a b - b a + /

d) a b 10 * /

e) 4 2 6 12 + * + 7 4 - 2 3 + * /

EXERCISE 3

WHICH STACK OPERATIONS LEAD TO
THE FOLLOWING NOTATIONS ?

a) (N1 \ N2 -- N2)

b) (N1 \ N2 -- N2 \ N1 \ N2)

c) (N1 \ N2 \ N3 -- N3 \ N2 \ N1)

d) (N1 \ N2 \ N3 -- N1 \ N2 \ N1)

e) (N1 \ N2 \ N3 -- N1 \ N1 \ N1 \ N2 \ N3)

EXERCISE 4

COMPLETE (N3 \ N2 \ N1 --
AFTER THE FOLLOWING STACK OPERATIONS:

a) ROT DUP

b) ROT ROT

c) SWAP ROT DUP

d) DUP SWAP ROT

e) SWAP DUP SWAP DUP

EXERCISE 5

WHICH NUMBER IS ON TOS AFTER
THE FOLLOWING OPERATIONS ?

a) `3 2 1 ROT + * 4 /`

b) `4 DUP * 12 -`

c) `3 4 5 ROT ROT OVER * * +`

d) `1 2 3 4 3 PICK + *`

e) `2 4 * 5 SWAP DUP ROT +`

EXERCISE 6

CALCULATE THE FOLLOWING EXPRESSIONS FOR ANY X LYING ON TOS :

a) $X^2 - X$

b) $X^2 - 4X + 2$

c) $X^3 - X + 2$

d) $5X^4 + 3X^2 + 1$

e) $X^3 - 5X^2 - 7X$

EXERCISE 7

CALCULATE THE FOLLOWING EXPRESSIONS
FOR ANY A, X LYING ON STACK
(X ON TOS) :

a) X / A

b) A / X

c) (A - X) / 7

d) 5 / (X - A)

e) (A - X) / (A + X)

SOLUTION 1

a) 4 3 - 6 2 + *

b) 5 7 - 3 8 3 - * +

c) 4 2 6 12 + * + 7 4 - 2 3 + * /

d) a b + c *

e) 3 a * b - 4 / c +

SOLUTION 2

a) (2 * 8) / (4 + 2)

b) ((1 - 2) * 3 + 7) / 2

c) (a - b) / (b + a)

d) a / (10 * b)

e) ((6 + 12) * 2 + 4) / ((7 - 4) * (2 + 3))

SOLUTION 3

a) SWAP DROP

b) SWAP OVER

c) SWAP ROT

d) DROP OVER

e) 2 PICK ROT ROT 2 PICK ROT ROT

SOLUTION 4

a) N3\N2\N1 -- N2\N1\N3\N3

b) N3\N2\N1 -- N1\N3\N2

c) N3\N2\N1 -- N1\N2\N3\N3

d) N3\N2\N1 -- N3\N1\N1\N2

e) N3\N2\N1 -- N3\N1\N2\N2\N2

SOLUTION 5

a) 2

b) 4

c) 41

d) 15

e) 13

SOLUTION 6

a)

	DUP	DUP	*	SWAP	-
STACK X	X	X	X^2	X	X^2 - X
-	X	X	X	X^2	-
-	-	X	-	-	-

b)

	DUP	DUP	*	SWAP	4	*	-	2	+
STACK X	X	X	X^2	X	4	4X	X^2 - 4X	2	X^2 -4X + 2
-	X	X	X	X^2	X	X^2	-	X^2 - 4X	-
-	-	X	-	-	X^2	-	-	-	-

SOLUTION 6

c)

	DUP	DUP	DUP	*	*	SWAP	-	2	+
STACK	X	X	X	X	X^2	X^3	X	$X^3 - X$	2	$X^3 - X + 3$
	-	X	X	X	X	X	X^3	-	$X^3 - X$	-
	-	-	X	X	X	-	-	-	-	-
	-	-	-	X	-	-	-	-	-	-

SOLUTION 6

d)

	DUP	DUP	DUP	DUP	*	*	*	5	*	SWAP	DUP	*
STACK	X	X	X	X	X	X^2	X^3	X^4	5	$5X^4$	X	X	X^2
	-	X	X	X	X	X	X	X	X^4	X	$5X^4$	X	$5X^4$
	-	-	X	X	X	X	X	-	X	-	-	$5X^4$	-
	-	-	-	X	X	X	-	-	-	-	-	-	-
	-	-	-	-	X	-	-	-	-	-	-	-	-

d continued)

	3	*	+	1	+
STACK	3	$3X^2$	$5X^4 + 3X^2$	1	$5X^4 + 3X^2 + 1$
	X^2	$5X^4$	-	$5X^4 + 3X^2$	-
	$5X^4$	-	-	-	-

SOLUTION 6

e)

	DUP	DUP	DUP	*	*	SWAP	DUP	DUP	*	5	*	SWAP	7	*
X	X	X	X	X^2	X^3	X	X	X	X^2	5	$5X^2$	X	7	7X
-	X	X	X	X	X	X^3	X	X	X	X^2	X	$5X^2$	X	$5X^2$
-	-	X	X	X	-	-	X^3	X	X^3	X	X^3	X^3	$5X^2$	X^3
-	-	-	X	-	-	-	-	X^3	-	X^3	-	-	X^3	-

e continued)

+	-
$5X^2 + 7X$	$X^3 - (5X^2 + 7X)$
X^3	-

SOLUTION 7

a)

```
SWAP /
```

b)

```
/
```

c)

```
- 7 /
```

d)

```
SWAP - 5 SWAP /
```

e)

	DUP	ROT	DUP	ROT	-	ROT	ROT	+	/
STACK	X	X	A	A	X	A-X	X	A	A+X	(A-X)/(A+X)
	A	X	X	A	A	A	A-X	X	A-X	
	-	A	X	X	A	X	A	A-X	-	
	-	-	-	X	X	-	-	-	-	

FORTH DEFINITIONS

SUMMARY

- GENERAL
- HOW TO DEFINE NEW WORDS
- MODULAR CODING

GENERAL

IN FORTH, ALL COMMANDS ARE CALLED **WORDS**. EACH **WORD** IS EXECUTED BY SIMPLY CALLING ITS NAME (IF THE WORD NEEDS PARAMETERS THEY ARE EXPECTED TO BE ON THE PARAMETER STACK BEFORE THE WORD IS CALLED).

EXAMPLES:

```
4 15 SPACES . CR

----------------4 OK
```

SPACES (NUMBER --)

THE WORD **SPACES** MOVES THE CURSOR TO THE RIGHT BY A NUMBER OF POSITIONS TAKEN FROM THE STACK.

GENERAL

EXAMPLES:

```
." I AM LEARNING FORTH"  CR

I AM LEARNING FORTH  OK
```

THE WORD ." OUTPUTS THE FOLLOWING STRING. IT MUST BE SEPERATED FROM THE STRING BY AT LEAST ONE BLANK.

STRING: I AM LEARNING FORTH

DELIMETER OF THE STRING: " (THIS IS NOT A FORTH WORD)

NOTE:

NAMES OF **FORTH WORDS** MAY CONSISTS OF UP TO 31 CHARACTERS.
THEY MUST BE SEPERATED FROM ONE ANOTHER BY AT LEAST ONE BLANK.

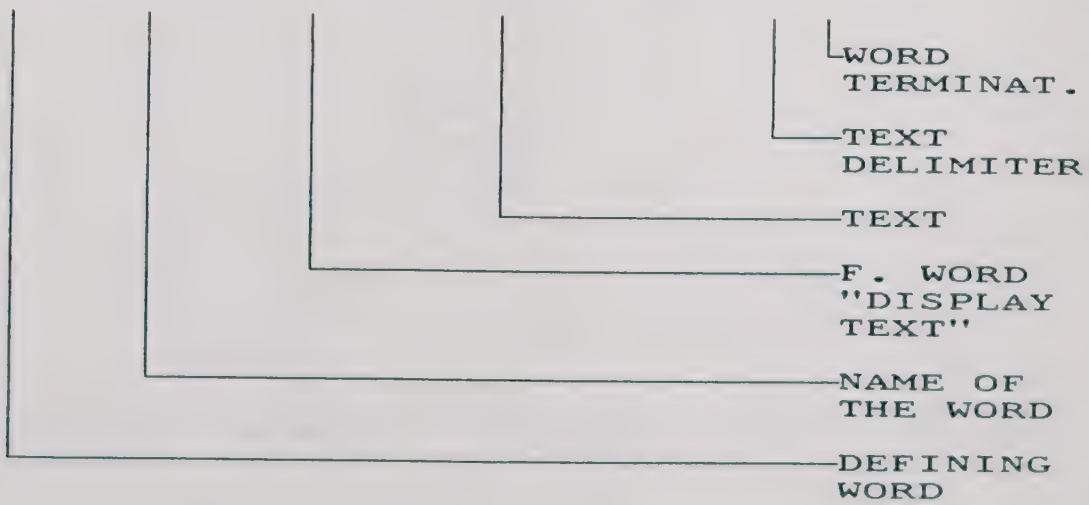
HOW TO DEFINE NEW WORDS

FORTH IS AN EXTENSIBLE LANGUAGE

YOU MAY USE ALREADY DEFINED WORDS TO EXTEND THE DICTIONARY BY DEFINING NEW WORDS.

EXAMPLE:

```
: NEW_WORD ." LEARNING FORTH" ;
```

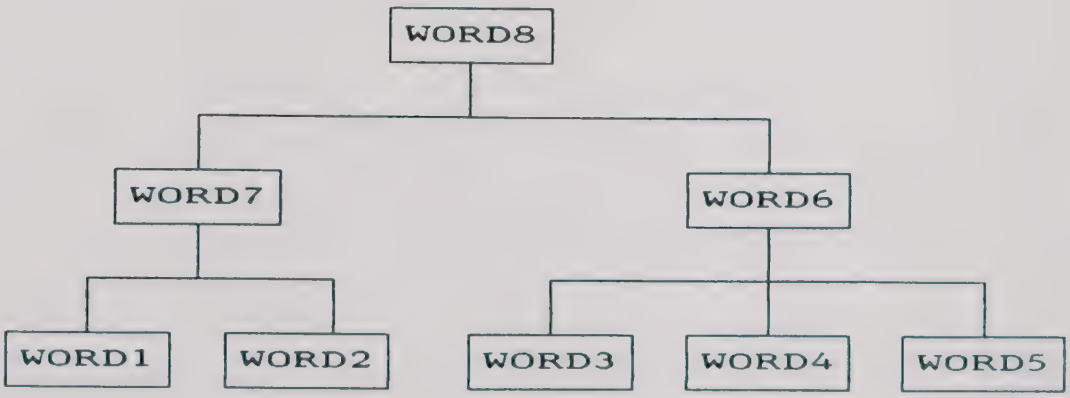


MODULAR CODING

FORTH ENCOURAGES MODULAR CODING.

A NEW **WORD** IS DEFINED BY USING PREVIOUSLY DEFINED WORDS.

SO YOU CAN DEFINE "COMPLICATED" **WORDS** BY "LESS COMPLICATED" **WORDS** AND THESE BY EVEN LESS COMPLICATED **WORDS**.



MODULAR CODING

BUT NOTE:

A **WORD** MUST BE DEFINED BEFORE IT IS CALLED OR USED IN A DEFINITION.

SO, THE DESIGN OF A **FORTH PROGRAM** MAY BE TOP-DOWN, FROM GENERAL TO SPECIAL **WORDS**, BUT THE DEFINITION OF THE **WORDS** IN A FORTH PROGRAM, MUST BE REVERSE FROM SPECIAL TO GENERAL.

EXERCISE 1

WORD DEFINITIONS :

a) DEFINE A WORD M5 WHICH MULTIPLIES THE VALUE ON STACK BY 5 AND PRINTS THE PRODUCT.

b) DEFINE A WORD -ROT WHICH ROTATES THE UPPER 3 ELEMENTS ON THE STACK IN THE OPPOSITE DIRECTION AS ROT.

c) DEFINE A WORD AREA WHICH EXPECTS THE 2 VALUES A AND B ON THE STACK AND PRINTS THE PRODUCT P = A * B .

d) DEFINE A WORD % WHICH EXPECTS THE 2 VALUES A AND X ON THE STACK AND PRINTS THE PERCENTAGE P = A * X / 100 .

e) DEFINE A WORD SEC WHICH EXPECTS THE 3 VALUES H, M AND S ON THE STACK AND PRINTS THE TOTAL NUMBER OF SECONDS.
TS = H * 3600 + M * 60 + S

EXERCISE 2

TEMPERATURE CONVERSION

A) CONVERT THE TEMPERATURE GIVEN IN DEGREES CELSIUS
TO DEGREES FAHRENHEIT. THE FORMULA IS :
T[FAHR] = 9/5 * T[CELSIUS] + 32

B) DEFINE A WORD WHICH CONVERTS FROM FAHRENHEIT
TO CELSIUS.

C) IMPROVE YOUR CONVERTING WORDS BY MAKING
THE FOLLOWING OUTPUT :
THE TEMPERATURE IS n DEGREES CELSIUS
EQUIVALENT TO m DEGREES FAHRENHEIT

EXERCISE 3

MONEY CHANGE

```
A QUARTER IS  25  PENNIES
A DIME IS     10  PENNIES
A NICKEL IS    5  PENNIES
```

DEFINE A WORD CHANGE WHICH ACCEPTS A NUMBER OF PENNIES AND PRINTS THE EQUIVALENT NUMBER OF QUARTERS DIMES NICKELS AND PENNIES.

EXERCISE 4

CREATE BANNERS

A) CREATE A WORD MARGIN WHICH EXPECTS A NUMBER ON THE STACK AND POSITIONS THE CURSOR TO THE SPECIFIED COLUMN OF THE NEXT LINE.

B) CREATE A WORD LM WHICH SETS THE LEFT MARGIN OF THE FOLLOWING OUTPUT TO COLUMN 35 OF THE NEXT LINE

C) DEFINE A NUMBER OF WORDS P0, P1, ... PE, PF WHICH DISPLAY THE BINARY REPRESENTATION OF A HEX DIGIT EACH.
(e.g. HEX "6" --> ".**.", HEX "B" --> "*.**")
USE LM TO DEFINE THE LEFT MARGIN FOR THE DISPLAY

D) DEFINE A NUMBER OF WORDS EACH OF WHICH DISPLAYS A LARGE CHARACTER AT THE SCREEN. USE THE WORDS P0 ... PF TO SETUP THE CHARACTER MATRIX.
NAME THE WORDS ACCORDING TO THE CHARACTER WHICH THEY STAND FOR.

E) DEFINE A WORD BANNER WHICH USES THE PREVIOUSLY DEFINED LARGE CHARACTERS TO DISPLAY A MESSAGE AT THE SCREEN.

SOLUTION 1

a)
```
: M5    ( N -- )
  5 * CR .
;
```

b)
```
: -ROT  ( N1\N2\N3 -- N3\N1\N2 )
  ROT ROT
;
```

c)
```
: AREA  ( A\B -- )
  * .
;
```

d)
```
: %     ( A\X -- )
  * 100 / .
;
```

e)
```
: SEC   ( H\M\S -- )
  ROT 3600 *
  ROT 60 *
  + + .
;
```

SOLUTION 2

a)
```
: T_FAHR     ( CELS -- FAHR )
  9 * 5 /  32  +
;
```

b)
```
: T_CEL      ( FAHR -- CELS )
  32 - 5 * 9 /
;
```

c)
```
: T_FAHRI    ( CELS -- )
  ." THE TEMPERATURE IS "  S.
  ."  DEGREES CELSIUS" CR
  ." EQUIVALENT TO "  T_FAHR .
  ."  DEGREES FAHRENHEIT"
;
```

SOLUTION 3

```
: CHANGE  ( PENNIES -- )
  25 /MOD . ."  QUARTERS" CR
  10 /MOD . ."  DIMES" CR
   5 /MOD . ."  NICKELS" CR
          . ."  PENNIES"
;
```

SOLUTION 4

a)
```
: MARGIN   ( N -- )
  CR SPACES
;
```

b)
```
: LM       ( -- )
  35 MARGIN
;
```

c)
```
: P0       ( -- )
  LM ."      "
;
: P1       ( -- )
  LM ."     *"
;
: P2       ( -- )
  LM ."    * "
;
: P3       ( -- )
  LM ."    **"
;
: P4       ( -- )
  LM ."   *  "
;
: P5       ( -- )
  LM ."   * *"
;
: P6       ( -- )
  LM ."   ** "
;
: P7       ( -- )
  LM ."   ***"
;
```

```
: P8       ( -- )
  LM ." *    "
;
: P9       ( -- )
  LM ." *   *"
;
: PA       ( -- )
  LM ." * * "
;
: PB       ( -- )
  LM ." * **"
;
: PC       ( -- )
  LM ." **   "
;
: PD       ( -- )
  LM ." ** *"
;
: PE       ( -- )
  LM ." *** "
;
: PF       ( -- )
  LM ." ****"
;
```

SOLUTION 4

d)
```
: F         ( -- )
  PF P8 PF P8 P8 P0 P0
;

: O         ( -- )
  P6 P9 P9 P9 P6 P0 P0
;

: R         ( -- )
  PE P9 PE PA P9 P0 P0
;

: T         ( -- )
  PE P4 P4 P4 P4 P0 P0
;

: H         ( -- )
  P9 P9 PF P9 P9 P0 P0
;
```

e)
```
: BANNER    ( -- )
  F O R T H
;
```

FORTH DATA STRUCTURES

SUMMARY

- DATA SIZES
- DATA TYPES
- DATA OPERATIONS
- DICTIONARY ENTRIES
- TABLES AND STRINGS
- SINGLE CHARACTERS

DATA SIZES

BYTE:

- 8 BIT BYTE OR CHARACTER
  FORTH WORDS DEALING WITH BYTES
  NORMALLY BEGIN WITH "C"
  EXAMPLE: C@, C!

WORD:

- 16 BIT WORD
  FORTH WORDS DEALING WITH WORDS
  NORMALLY BEGIN WITH "W"
  EXAMPLE: W@, W!

LONG WORD:

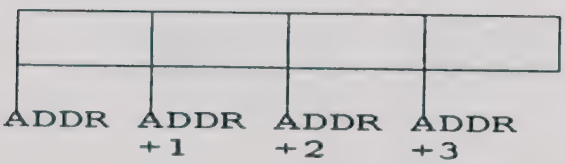
- 32 BITS (STANDARD SIZE IN THE K1197)

DATA SIZES

LONG WORD

WORD

BYTE



NOTE:

- ADDRESSES ARE 24 BIT ADDRESSES
- ADDRESSES OF WORDS AND LONG WORDS ARE EVEN

DATA TYPES

CONSTANTS:

- FIXED VALUES WHICH ARE PLACED ON THE STACK DIRECTLY BY CALLING THE NAME OF THE **CONSTANT**

VARIABLES:

- VALUES WHOSE ADDRESSES ARE PLACED ON THE STACK BY CALLING THE NAME OF THE **VARIABLE**

- THEY MAY BE USED FOR VECTORS, MATRICES BUFFERS ETC.

DATA TYPES

LITERALS:

- NUMERIC LITERALS, MAXIMUM OF 32 BITS
  THESE ARE DECIMAL NUMBERS
  EXAMPLES: 123, -15

- EXTENDED NUMERIC LITERALS IN **DECIMAL** MODE
  PREFIXES ARE:
  - 0X FOR HEX
  - 0C FOR OCTAL
  - 0B FOR BINARY
  EXAMPLES: 0X1FFF, 0C177, 0B11111

- SHORT IMMEDIATE ASCII LITERALS
  E.G. #" TEXT"
  WHERE **TEXT** MAY BE REPLACED BY UP TO 4 CHARACTERS

- LONG POINTER LITERALS
  E.G. " LONG TEXT"
  UP TO 255 CHARACTERS ARE ALLOWED

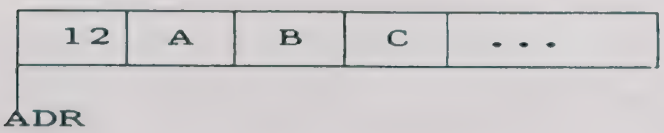
- LONG **HEX** POINTER LITERALS
  E.G. X" 03FF" WHERE HEX CHARACTER PAIRS ARE ENTERED.
  EVEN NUMBER OF CHARACTERS IS NOT TESTED!

DATA TYPES

STRINGS:

- TYPE 1 MAY BE ACCESSED BY ADDRESS AND EXPLICIT LENGTH
  E.G. ADR 12 TYPE

- TYPE 2 (PACKED STRING) MAY BE ACCESSED BY ADDRESS POINTING TO THE LENGTH BYTE FOLLOWED BY THE STRING



- TYPE 3 SHORT STRINGS OF MAX. 32 BITS PASSED ON THE STACK BY #''

NOTE:

PACKED STRINGS ARE LONG POINTER LITERALS.

DATA OPERATIONS

DEFINING A CONSTANT:

VALUE CONSTANT NAME

EXAMPLE:

```
12  CONSTANT  MAX_LENGTH
MAX_LENGTH .  CR

12 OK
```

NOTE:

CALLING A CONSTANT BY ITS NAME LEAVES THE VALUE OF THE CONSTANT ON TOS

DATA OPERATIONS

DEFINING A VARIABLE:

INITIAL VALUE **VARIABLE** NAME

EXAMPLE:

```
0  VARIABLE  COUNTER
COUNTER .  CR

220312 OK
```

NOTE:

CALLING A VARIABLE BY ITS NAME LEAVES THE ADDRESS OF THE VALUE ON TOS

DATA OPERATIONS

FETCHING AND STORING VALUES:

@ (pronounce FETCH)

@ (ADDRESS -- CONTENTS OF ADDRESS)

EXAMPLE:

```
COUNTER @ .  CR
0 OK
```

DATA OPERATIONS

FETCHING AND STORING VALUES:

! (pronounce STORE)

! (VALUE\ADDRESS --)

EXAMPLE:

```
1 COUNTER !
COUNTER @ .  CR
```

1 OK

DATA OPERATIONS

FETCHING AND STORING VALUES:

+! (pronounce PLUS-STORE)

```
+! ( D\ADDRESS --  )
INCREASES CONTENTS OF ADDRESS BY D
```

EXAMPLE:

```
1 COUNTER +!
COUNTER @ .  CR

2 OK
```

NOTE:

@, ! AND +! EXPECT EVEN ADDRESSES!
32 BIT VALUES ARE FETCHED AND STORED

DATA OPERATIONS

FETCHING AND STORING 16 BIT VALUES (WORDS):

W@, W!

USE: LIKE @ AND !

FETCHING AND STORING 8 BIT VALUES (BYTES):

C@, C!

USE: LIKE @ AND !

NOTE:

W@, AND W! NEED EVEN ADDRESSES!

DATA OPERATIONS

FORTH WORD DP:

THE FORTH WORD DP (DICTIONARY POINTER) POINTS TO THE NEXT AVAILABLE DICTIONARY SPACE.

IT MAY BE ACCESSED BY:

DP @

OR EQUIVALENT BY

HERE

THE DICTIONARY POINTER IS INCREASED BY EACH DEFINING WORD WHICH MAKES A DICTIONARY ENTRY (E.G. CONSTANT, VARIABLE, :) AND BY **ALLOT** WHICH RESERVES BUFFER SPACE FOR A FIELD.

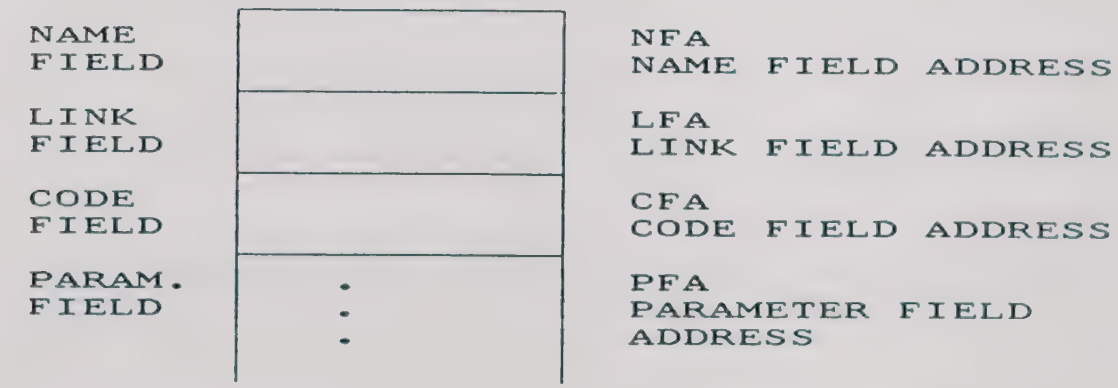
EXAMPLE:

```
0 VARIABLE BUFF 80 ALLOT
```

DICTIONARY ENTRIES

STRUCTURE OF A DICTIONARY ENTRY:

EACH DICTIONARY ENTRY HAS THE FOLLOWING STRUCTURE:



THE LENGTH OF THE NAME FIELD IS VARIABLE BUT LIMITED TO 31 CHARACTERS.

LINK FIELD, AND CODE FIELD HAVE CONSTANT LENGTH FOR EACH DICTIONARY ENTRY (32 BITS EACH).

THE LENGTH OF THE PARAMETER FIELD IS VARIABLE.

DICTIONARY ENTRIES

STRUCTURE OF A DICTIONARY ENTRY:

THE NAME FIELD CONTAINS THE NAME OF THE DICTIONARY ENTRY, E.G.

- NAME OF A VARIABLE
- NAME OF A CONSTANT
- NAME OF A **FORTH WORD** DEFINED BY ":"

THE LINK FIELD CONTAINS THE NAME FIELD ADDRESS (NFA) OF THE PREVIOUS DICTIONARY ENTRY.

THE CODE FIELD POINTS TO THE CODE OF "WHAT TO DO WITH THE PARAMETER IN THE PARAMETER FIELD"

DICTIONARY ENTRIES

STRUCTURE OF A DICTIONARY ENTRY:

EXAMPLES:

1) DICTIONARY ENTRY: CONSTANT

THE CFA POINTS TO THE CODE
"PUT THE VALUE AT PFA ON THE STACK".
THE VALUE OF THE CONSTANT IS STORED AT PFA AT COMPILE TIME.

2) DICTIONARY ENTRY: VARIABLE

THE CFA POINTS TO THE CODE
"PUT THE PFA ON THE STACK".
THE CURRENT VALUE OF THE VARIABLE IS STORED AT PFA.

3) LOW LEVEL DEFINITIONS (PRIMITIVES)

THE CFA POINTS TO THE PFA.
THE PARAMETER FIELD CONTAINS EXECUTABLE MACHINE CODE.

DICTIONARY ENTRIES

STRUCTURE OF A DICTIONARY ENTRY:

EXAMPLES:

4) DICTIONARY ENTRY :

" : " DEFINES A **FORTH WORD** IN TERMS OF OTHER FORTH WORDS ALREADY EXISTENT IN THE DICTIONARY.

THE CFA POINTS TO THE CODE
"EXECUTE THE WORDS CORRESPONDING TO THE PFA-ENTRIES UNTIL " ; " IS REACHED".

THE PARAMETER FIELD CONTAINS AS MANY ENTRIES AS THERE ARE "OTHER WORDS" INCLUDING THE " ; ".
EACH ENTRY IS THE CFA OF THE CORRES-PONDING WORD.

DICTIONARY ENTRIES

VECTORED EXECUTION

' (PRONOUNCED "TICK") LEAVES THE PFA OF THE FOLLOWING **FORTH WORD** ON TOS

USE: : TEST ." HELLO" ;

' TEST

A **FORTH WORD** WHOSE PFA IS ON TOS CAN BE EXECUTED BY:

CFA EXECUTE

EXAMPLE: ' TEST **CFA EXECUTE** CR

HELLO OK

CFA CONVERTS THE PFA INTO THE CFA AND **EXECUTE** EXECUTES THE WORD.

THIS IS OFTEN USED, WHEN A TABLE IS FILLED WITH PFA'S OF FORTH WORDS AND A WORD IS EXECUTED ACCORDING TO A CURRENT OFFSET.

TABLES AND STRINGS

INITIALIZING A TABLE:

WE HAVE SEEN, THAT BUFFER SPACE FOR A TABLE OR AN ARRAY CAN BE RESERVED BY

```
0  VARIABLE  BUFF 80 ALLOT
```

BUFF IS INITIALIZED IN ITS 4 LOWEST BYTES WITH ZERO, BUT THE CONTENTS OF THE REST IS STILL UNDEFINED.

```
0  VARIABLE BUFF 1 , 2 , 3 , 4 , 5 ,
```

WILL DEFINE BUFFER SPACE OF 6 LONG WORDS = 24 BYTES AND INITIALIZE THE BUFFER WITH THE VALUES 0...5.

, (PRONOUNCED "COMMA") STORES THE VALUE ON TOS AT HERE AND INCREASES THE DICTIONARY POINTER BY 4.

NOTE:

THE 16 BIT AND 8 BIT VERSIONS OF , ARE W, AND C,

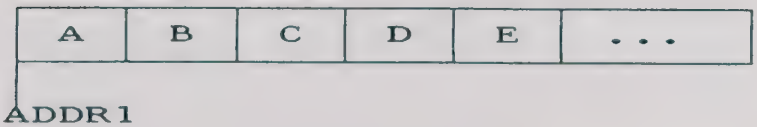
TABLES AND STRINGS

MOVING CONTENTS OF BUFFER SPACES

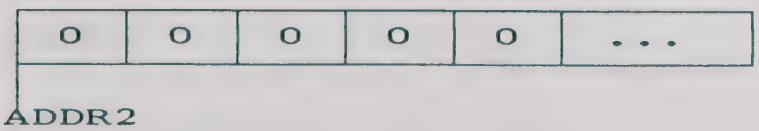
CMOVE (ADDR1\ADDR2\N --)

MOVES N BYTES BEGINNING AT ADDR1 TO ADDR2

EXAMPLE:



BEFORE CMOVE

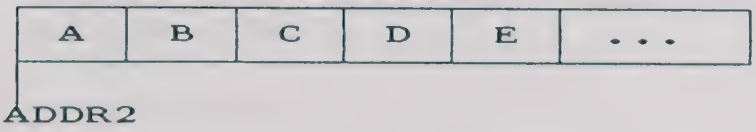


TABLES AND STRINGS

MOVING CONTENTS OF BUFFER SPACES

```
ADDR1 ADDR2 5 CMOVE
```

AFTER CMOVE



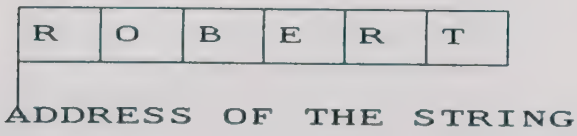
NOTES:

- ADDR1 AND ADDR2 WILL USUALLY BE NAMES OF VARIABLES (EVENTUALLY PLUS AN OFFSET)
- THE 16 BIT VERSION OF CMOVE IS MOVE IN THIS CASE ADDR1 AND ADDR2 MUST BE EVEN NUMBERS

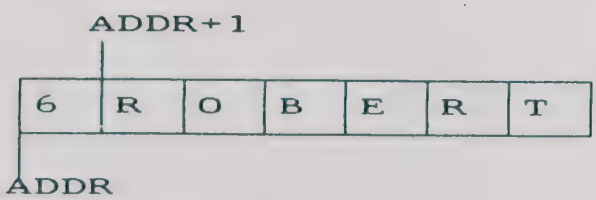
TABLES AND STRINGS

STRINGS

A STRING IS A CHAIN OF CHARACTERS IN BUFFER SPACE.



A STRING, WHERE THE VALUE OF THE FIRST CHARACTER GIVES THE NUMBER OF CHARACTERS OF THE REST OF THE STRING IS CALLED A PACKED STRING.

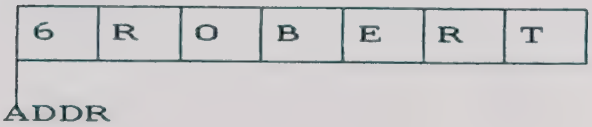


TABLES AND STRINGS

STRINGS

IT IS OFTEN USEFUL TO OPERATE WITH PACKED STRINGS.

THE SEQUENCE " ROBERT" CREATES A PACKED STRING AND LEAVES THE ADDRESS ADDR ON TOS.



TABLES AND STRINGS

STRINGS

TO "UNPACK" A PACKED STRING, THERE IS THE FORTH WORD **COUNT**.

```
COUNT   ( ADDR -- ADDR+1\N )
```

COUNT INTERPRETS THE VALUE OF THE BYTE AT ADDRESS ADDR AS BYTE COUNT AND LEAVES IT ON TOS.

EXAMPLE:

```
" ROBERT" COUNT .  CR

6 OK
```

TABLES AND STRINGS

STRINGS

COUNT LEAVES THE NECESSARY STACK PARAMETERS FOR THE STRING OUTPUT COMMAND

- STYPE (CHAR ADDR.\CHAR COUNT\ --)

EXAMPLES:

```
" ROBERT" COUNT STYPE  CR
ROBERT OK

" ROBERT" 1+ 4 STYPE  CR
ROBE OK
```

TABLES AND STRINGS

STRINGS

STYPE WITH ATTRIBUTES: **STYPE_A**

POSSIBLE ATTRIBUTES ARE FOREGROUND OR BACKGROUND COLORS AND CHARACTER CODE ACCORDING TO THE FOLLOWING TABLE:

FORE GROUND	BACK GROUND
WHI_FG = XXXXXX00	
	RED_BG = XXXX11XX
GRN_FG = XXXXXX01	
	BLU_BG = XXXX01XX
YEL_FG = XXXXXX10	
	MAG_BG = XXXX10XX
CYA_FG = XXXXXX11	
	BLK_BG = XXXX00XX

REV_VIDEO	= 0001XXXX
ASCII	= 0000XXXX
EBCDIC	= 0010XXXX
HEX SET	= 0100XXXX
TTX	= 0110XXXX
JS8	= 1000XXXX

X = DON'T CARE

0 = CLEARED

1 = SET

TABLES AND STRINGS

STRINGS

THE FORMAT OF STYPE_A IS:

```
STYPE_A
( CHAR ADDR.\CHAR COUNT\ATTRIBUTE -- )
```

EXAMPLE:

```
" ROBERT" COUNT RED_BG STYPE_A CR
ROBERT OK   (RED BACKGROUND)
```

TABLES AND STRINGS

STRINGS

ANY SEQUENCE LIKE

" ROBERT"

WILL PUT THE STARTING ADDRESS OF THE PACKED STRING ON TOS.

THE ACTUAL STRING IS AUTOMATICALLY STORED IN A SO-CALLED **LITERAL BUFFER.**

THERE ARE 8 ASCII AND 8 HEX LITERAL BUFFERS AVAILABLE IN THE SYSTEM. THE LITERAL BUFFERS ARE USED "ROUND ROBIN".

THEREFORE, IF A USER NEEDS MORE THAN 8 PACKED STRINGS AT A TIME, HE HAS TO DEFINE HIS OWN BUFFERS (USING VARIABLE AND ALLOT) AND MOVE THE PACKED STRINGS IN THOSE BUFFERS.

TABLES AND STRINGS

STRINGS

EXAMPLE:

```
0 VARIABLE MY-BUFFER 80 ALLOT  CR

" TESTSTRING" DUP C@ 1+
MY-BUFFER SWAP CMOVE  CR

MY-BUFFER COUNT STYPE  CR
```

TESTSTRING OK

TABLES AND STRINGS

STRINGS

SOMETIMES, A RUNNING PROGRAM EXPECTS A STRING FROM THE USER BY TERMINAL INPUT.

QUERY IS THE PROPER **FORTH WORD** TO EXPECT A STRING TERMINATED BY CR AND SAVE IT IN THE TERMINAL INPUT BUFFER (TIB).

THE SEQUENCE **c WORD** INVESTIGATES TIB FOR A CHARACTER WITH ASCII VALUE **c** AND STORES THE PACKED STRING DELIMITED BY **c** AT ADDRESS HERE.

SINGLE CHARACTERS

OUTPUT OF A SINGLE CHARACTER

EMIT (C --)

EMIT DISPLAYS THE ASCII REPRESENTATION OF THE VALUE ON TOS.

EXAMPLE:

```
65 EMIT  CR
A OK
```

SINGLE CHARACTERS

INPUT OF A SINGLE CHARACTER

KEY (-- C)

KEY EXPECTS THE INPUT OF A SINGLE CHARACTER FROM KEYBOARD; THE ASCII VALUE OF THE CHARACTER IS LEFT ON TOS.

EXAMPLE: (PAUSE-FUNCTION)

```
: PAUSE KEY DROP ;
```

PROGRAM EXECUTION STOPS UNTIL A KEY IS PRESSED.
THIS SIMPLE WORD CAN BE USEFUL IN DEBUGGING.

EXERCISE 1

VARIABLES

A) DEFINE A VARIABLE "SWITCH" WITH DEFAULT VALUE 1.

B) DEFINE THE WORDS "S_ON" AND "S_OFF" WHICH SET THE VARIABLE SWITCH TO 1 OR 0 RESPECTIVELY.

C) DEFINE THE WORD "S?" WHICH PRINTS THE CURRENT CONTENTS OF THE VARIABLE SWITCH IN THE FORM:
SWITCH = n

EXERCISE 2

VARIABLES AND CONSTANTS

A) DEFINE A VARIABLE "COUNTER" WITH DEFAULT VALUE 0.

B) DEFINE A CONSTANT "INC" WITH VALUE 15.

C) DEFINE A WORD "C?" WHICH PRINTS THE CURRENT CONTENTS OF "COUNTER" IN THE FORM:
COUNTER = n

D) DEFINE THE WORDS "C+" AND "C-" WHICH INCREMENT / DECREMENT THE CONTENTS OF "COUNTER" BY THE VALUE OF "INC" AND PRINT ITS NEW VALUE.

EXERCISE 3

POINTERS

A) DEFINE A NUMBER OF VARIABLES "A, B, C, ..." WITH DEFAULT VALUE 0.

B) DEFINE A VARIABLE "PTR" WITH DEFAULT VALUE "address of "A" ".

C) DEFINE A WORD "P!" WHICH STORES THE VALUE ON STACK IN THE LOCATION WHERE "PTR" POINTS TO.

D) DEFINE A WORD "P@" WHICH PRINTS THE CONTENTS OF THE LOCATION WHERE "PTR" POINTS TO.

E) DEFINE A WORD "POINT" WHICH STORES THE ADDRESS ON THE STACK IN THE VARIABLE "PTR".

F) MAKE "PTR" POINT TO DIFFERENT LOCATIONS (A, B, C, ...) .
STORE AND FETCH VALUES TO AND FROM THESE LOCATIONS USING THE WORDS "P!" AND "P@".

EXERCISE 4

ARRAYS

A) DEFINE AN ARRAY "L_ARRAY" FOR 5 LONGWORDS (4 BYTES EACH).

B) DEFINE A WORD "PUT" WHICH EXPECTS THE 2 VALUES V AND N AND STORES THE VALUE V AT THE ELEMENT N OF THE ARRAY. (N = 0, 1, ... 4)

C) DEFINE A WORD "GET" WHICH EXPECTS N ON THE STACK AND PRINTS THE CONTENTS OF THE ELEMENT N OF "L_ARRAY"

D) IMPROVE YOUR WORDS "PUT" AND "GET" BY A RANGE CHECK FOR THE ELEMENT NUMBER N.

HINT: USE THE WORDS "MIN" AND "MAX" TO DO THE RANGE CHECKING.

EXERCISE 5

ARRAYS

A) DEFINE AN ARRAY "A_HUNDRED" AND PRESET IT WITH THE NUMBERS 0, 100, 200, ... 900 . DEFINE A WORD "HUNDRED" WHICH EXPECTS AN OFFSET (0 - 9) AND PRINTS THE CORRESPONDING NUMBER OF "A_HUNDRED".

B) DEFINE AN ARRAY "A_CHAR" AND USE "C," TO PRESET IT WITH THE ASCII VALUES OF THE CHARACTERS A THROUGH E. (65, 66, ...) DEFINE A WORD "CHAR" WHICH EXPECTS AN OFFSET (0 - 4) ON TOS AND PRINTS THE CORRESPONDING CHARACTER.
HINT: USE THE WORD "EMIT (C --)" TO PRINT THE CHARACTER.

EXERCISE 6

ARRAYS

A) DEFINE AN ARRAY "TAB" WHICH ALLOCATES MEMORY SPACE FOR 20 BYTES.
DEFINE A VARIABLE "OFFSET" WHICH CONTAINS THE CURRENT OFFSET INTO THE TABLE "TAB"
(DEFAULT OFFSET = 0)

B) DEFINE A WORD "NEXT" WHICH INCREMENTS "OFFSET" BY 1 EACH TIME IT IS CALLED.
IF "OFFSET" GETS GREATER THAN 19, "NEXT" SHOULD SET IT BACK TO 0 .

C) DEFINE A WORD "T_FILL" WHICH EXPECTS A NONNEGATIVE NUMBER ON TOS AND STORES THE NUMBER MODULO 256 AT THE NEXT AVAILABLE BYTE OF "TAB".

D) DEFINE A WORD "TAB?" WHICH EXPECTS AN OFFSET (0 - 19) AND PRINTS THE CORRESPONDING VALUE OF "TAB".

EXERCISE 7

STRINGS

A) DEFINE 3 BUFFERS X, Y, Z FOR 82 BYTES EACH.

B) DEFINE A WORD "NOTE" WHICH EXPECTS A PACKED STRING AND THE ADDRESS OF ONE OF THE BUFFERS X, Y, Z AND WHICH STORES THE PACKED STRING INTO THIS BUFFER.

C) DEFINE A WORD "SHOW" WHICH EXPECTS THE ADDRESS OF A PACKED STRING AND DISPLAYS THE STRING AT THE SCREEN WITH RED BACKGROUND.

SOLUTION 1

VARIABLES

A)

```
1 VARIABLE SWITCH
```

B)

```
: S_ON   ( -- )
  1 SWITCH !
;

: S_OFF  ( -- )
  0 SWITCH !
;
```

C)

```
: S?     ( -- )
  CR ." SWITCH = "
  SWITCH @ .
;
```

SOLUTION 2

VARIABLES AND CONSTANTS

A)
```
0 VARIABLE COUNTER
```

B)
```
15 CONSTANT INC
```

C)
```
: C?      ( -- )
  CR ." COUNTER = "
  COUNTER @ .
;
```

D)
```
: C+      ( -- )
  INC COUNTER +! C?
;

: C-      ( -- )
  COUNTER  @ INC - COUNTER ! C?
;
```

SOLUTION 3

POINTERS

A)
```
0 VARIABLE A
0 VARIABLE B
0 VARIABLE C
```

B)
```
A VARIABLE PTR
```

C)
```
: P!      ( VALUE -- )
  PTR @ !
;
```

D)
```
: P@      ( -- )
  PTR @ @ CR .
;
```

E)
```
: POINT   ( ADDR -- )
  PTR !
;
```

SOLUTION 4

ARRAYS

A)
```
0 VARIABLE L_ARRAY 16 ALLOT
```

B)
```
: PUT          ( V\N -- )
  4 * L_ARRAY + !
;
```

C)
```
: GET          ( N -- )
  4 * L_ARRAY + @ CR .
;
```

D)
```
: CHECK        ( V\N -- V\M )
  4 MIN 0 MAX
;

: IPUT         ( V\N -- )
  CHECK PUT
;

: IGET         ( N -- )
  CHECK GET
;
```

SOLUTION 5

ARRAYS

A)
```
0 VARIABLE A_HUNDRED 100 , 200 , 300 ,
                     400 , 500 , 600 ,
                     700 , 800 , 900 ,

: HUNDRED          ( OFFSET -- )
  4 * A_HUNDRED + @ CR .
;
```

B)
```
0 VARIABLE A_CHAR   65 C, 66 C, 67 C, 68 C, 69 C,

: CHAR             ( OFFSET -- )
  4+ A_CHAR +  C@ EMIT
;
```

SOLUTION 6

ARRAYS

A)
```
0 VARIABLE TAB 16 ALLOT
0 VARIABLE OFFSET
```

B)
```
: NEXT          ( -- )
  OFFSET @ 1+ 20 MOD OFFSET !
;
```

C)
```
: T_FILL        ( N -- )
  256 MOD OFFSET @ TAB + C! NEXT
;
```

D)
```
: TAB?          ( OFFSET -- )
  TAB + C@ .
;
```

SOLUTION 7

STRINGS

A)
```
0 VARIABLE X 78 ALLOT
0 VARIABLE Y 78 ALLOT
0 VARIABLE Z 78 ALLOT
```

B)
```
: NOTE                  ( STR\BUF --              )

  OVER C@ 1+            ( STR\BUF\LEN+1           )
  CMOVE                 ( MOVE LEN+1 BYTES        )
                        ( FROM STR TO BUF         )
;
```

C)
```
: SHOW                  ( BUF --                  )

  COUNT RED_BG          ( BUF+1\LEN\ATTR          )
  STYPE_A               ( DISPLAY LEN BYTES       )
                        ( WITH RED BACKGROUND     )
;
```

FORTH CONTROL STRUCTURES

SUMMARY

- GENERAL
- CONDITIONAL BRANCH
- LOOPS
- LOOPS WITH UNDEFINED REPETITION NUMBER
- CASE CONSTRUCT

GENERAL

FORTH SUPPORTS STRUCTURED PROGRAMMING.

THE FOLLOWING CONTROL STRUCTURES ARE STRUCTURED PROGRAM BLOCKS.

EACH BLOCK HAS ONE ENTRY AND ONE EXIT.

GOTO'S OR JUMPS ARE NOT PERMITTED.

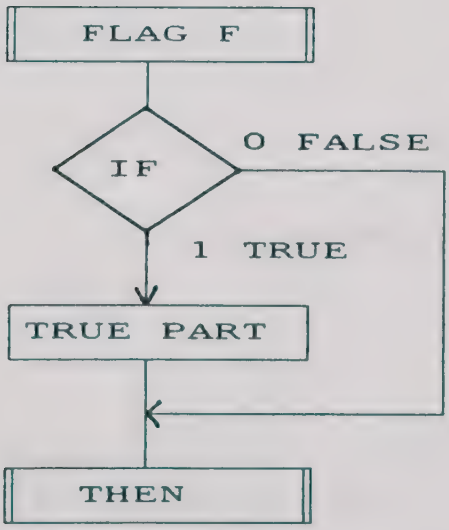
NOTE:

EACH OF THE FOLLOWING CONTROL STRUCTURES IS ONLY ALLOWED WITHIN A FORTH DEFINITION!

CONDITIONAL BRANCH

IF ... THEN CONSTRUCTION

ENTRY



F CAN BE CREATED BY ANY FORTH CODE, E.G. COMPARISONS

IF TESTS THE FLAG

IF F=0 OR "FALSE", THE PROGRAM CONTINUES AFTER THEN

IF F<>0 OR "TRUE", THE "TRUE PART" IS EXECUTED BEFORE CONTINUING AFTER THEN

EXIT

CONDITIONAL BRANCH

IF ... THEN CONSTRUCTION

EXAMPLE:

```
( NUMBER -- )

: NEGA 0<              ( FLAG CREATED )
   IF                  ( TRUE PART )
      ." THIS NUMBER IS NEGATIVE"
   THEN
;
```

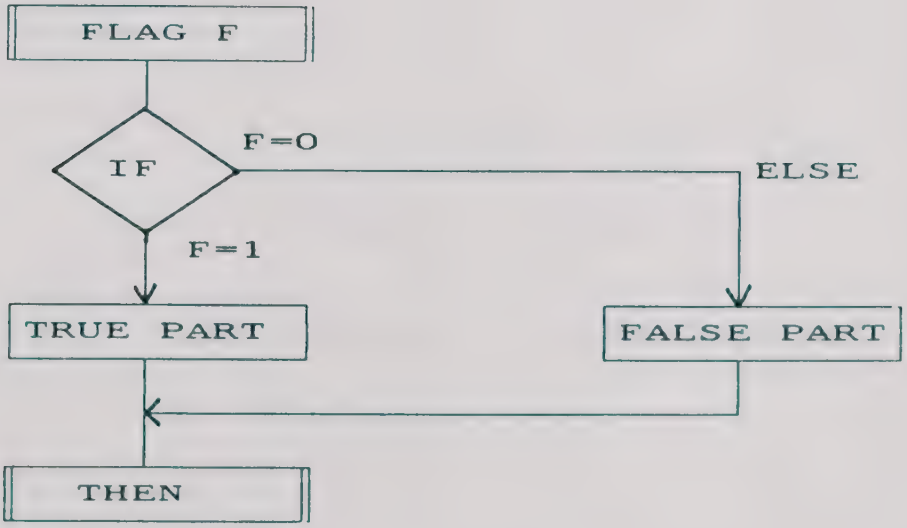
NOTES:

- IF CONSUMES THE FLAG
- THE "TRUE PART" MAY CONSIST OF ANY FORTH CODE, BUT EACH CONTROL STRUCTURE CONTAINED IN IT MUST BE CLOSED

CONDITIONAL BRANCH

IF ... ELSE ... THEN CONSTRUCTION

ENTRY



EXIT

THE **IF ELSE THEN** CONSTRUCTION WORKS LIKE **IF THEN** EXCEPT THAT IN F=0 THE "FALSE PART" IS EXECUTED BEFORE CONTINUING AFTER THEN.

CONDITIONAL BRANCH

IF ... ELSE ... THEN CONSTRUCTION

EXAMPLE:

```
( NUMBER -- )

: TEST1 0<              ( FLAG CREATED )
   IF                   ( TRUE PART )
      ." THIS NUMBER IS NEGATIVE"
   ELSE                 ( FALSE PART )
      ." THIS NUMBER IS NONNEGATIVE"
   THEN
;
```

NOTES:

- THE PREVIOUS NOTES APPLY TOO
- THE "FALSE PART" MAY ALSO CONTAIN ONLY CLOSED CONTROL STRUCTURES
- ELSE SEPERATES THE "TRUE PART" FROM THE "FALSE PART"

CONDITIONAL BRANCH

NESTING OF IF...ELSE...THEN CONSTRUCTION

THE IF...THEN AND IF...ELSE...THEN STRUCTURES MAY BE NESTED IN NEARLY ARBITRARY DEPTH.

EXAMPLE OF NESTING:

```
( NUMBER -- )

: TEST2
   DUP 0 >                  ( FLAG CREATED )
  ┌ IF
  │
  │    ." POSITIVE "
  │    5 >                  ( NEW FLAG CREATED )
  │  ┌ IF
  │  │  ." AND GREATER THAN 5"
  │  └ THEN
  │
  ├ ELSE
  │
  │    0=                   ( NEW FLAG CREATED )
  │  ┌ IF
  │  │  ." EQUAL TO ZERO"
  │  ├ ELSE
  │  │  ." NEGATIVE"
  │  └ THEN
  │
  └ THEN
;
```

LOOPS

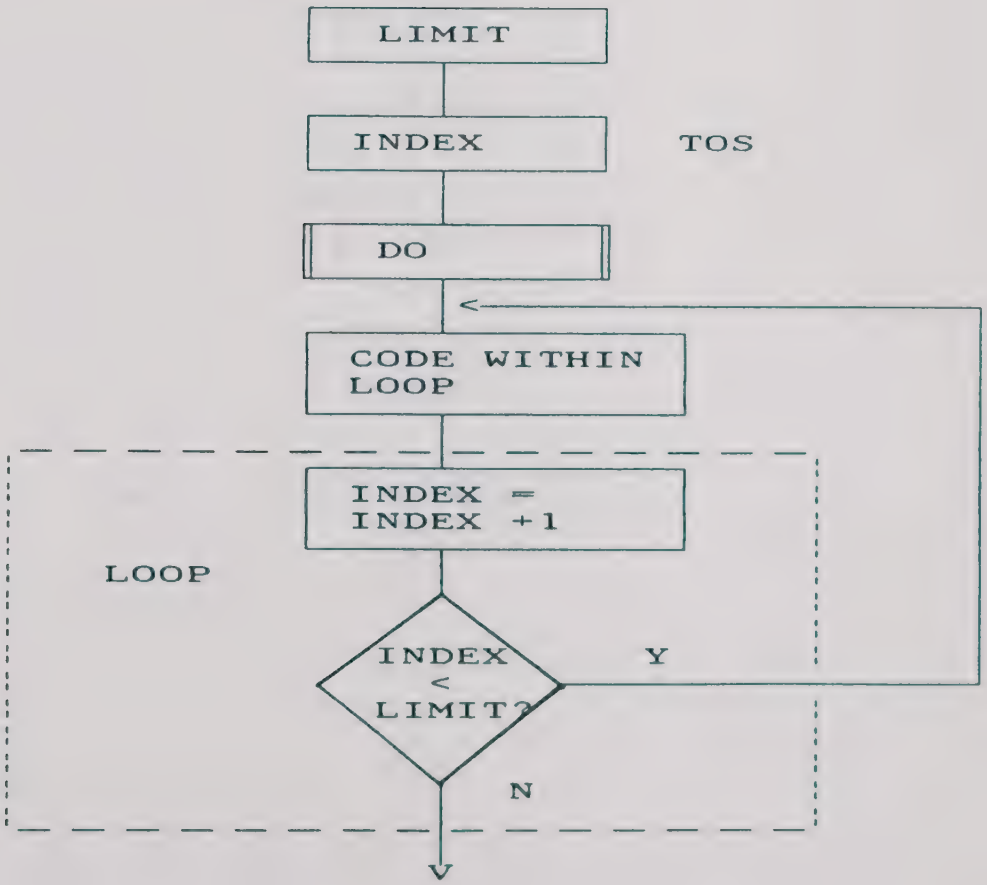
DO ... LOOP
DO ... +LOOP (INDUCTIVE LOOPS)

USED FOR REPETITION OF THE CODE BETWEEN DO AND LOOP OR +LOOP RESPECTIVELY.

THERE IS A LOOP COUNTER, WHICH MUST BE GIVEN A STARTING (INDEX) AND TERMINATING (LIMIT) VALUE.

LOOPS

DO ... LOOP



LOOPS

DO ... LOOP

EXAMPLE:

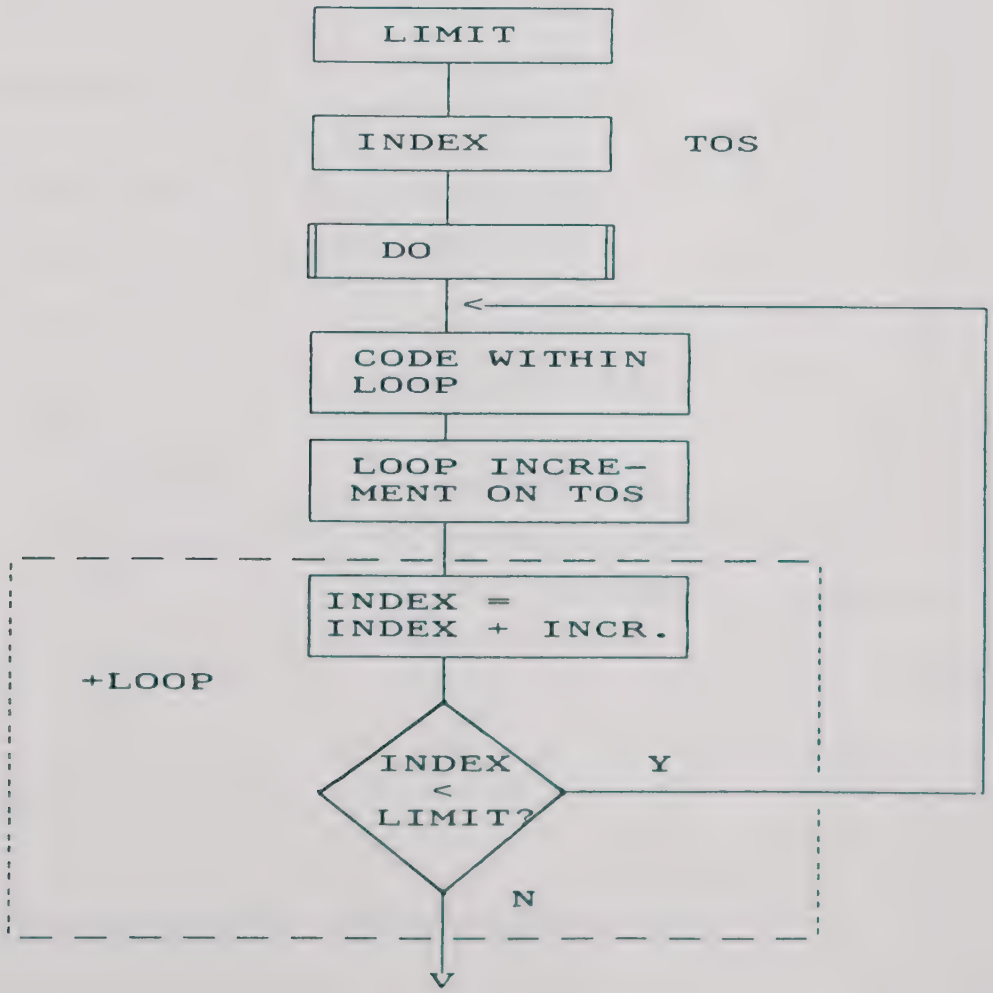
```
0 VARIABLE VAR

: TEST3 ( - )
   10          ( LIMIT )
   0           ( INDEX )
   DO
      VAR @ .     ( OUTPUT VAR )
      3 VAR +!    ( INCREASE VAR BY 3 )
   LOOP
;
```

USE:

```
TEST3  CR
0 3 6 9 12 15 18 21 24 27 OK
```

LOOPS

DO ... +LOOP



LOOPS

DO ... +LOOP

EXAMPLE:

```
0 VARIABLE VAS

: TEST4 ( - )
   30          ( LIMIT )
   0           ( INDEX )
   DO
      VAS @ .     ( OUTPUT VAS )
      3 VAS +!    ( INCREASE VAS BY 3 )
      3           ( LOOP COUNTER INCREMENT)
   +LOOP
;
```

USE:

```
TEST4   CR
0 3 6 9 12 15 18 21 24 27 OK
```

LOOPS

DO ... LOOP
DO ... +LOOP (INDUCTIVE LOOPS)

NOTES:

- THE FORTH WORDS WITHIN THE LOOP ARE EXECUTED AT LEAST ONCE SINCE THE TESTING OF THE TERMINATING CONDITION OCCURS AT THE END OF THE LOOP

- DO MOVES INDEX AND LIMIT FROM THE PARAMETER STACK TO THE RETURN STACK

- THE CURRENT VALUE OF THE LOOP COUNTER CAN BE ACCESSED BY THE FORTH WORD I WHICH LEAVES THE VALUE ON TOS

- IF THE LOOP INCREMENT IS POSITIVE, THE LOOP IS LEFT WHEN I IS GREATER THAN OR EQUAL TO THE TERMINATING VALUE
  IF THE INCREMENT IS NEGATIVE, THE LOOP IS LEFT WHEN I IS SMALLER THAN THE TERMINATING VALUE

- THE CODE WITHIN A LOOP MAY CONTAIN ONLY CLOSED CONTROL STRUCTURES

LOOPS

NESTING OF LOOPS

- LOOPS MAY BE NESTED IN NEARLY ARBITRARY DEPTH
- THE LOOP COUNTER OF THE INNERMOST LOOP CAN BE ACCESSED BY I
- THE LOOP COUNTER OF THE NEXT OUTER LOOP CAN BE ACCESSED BY J
- FURTHER LOOP COUNTERS MUST BE ADMINISTERED BY THE PROGRAMMER HIMSELF

EXAMPLE:

```
: IJ_LOOP
   4 0
   DO
      ." I =" I .   CR
      4 0
      DO
         ." I =" I . SPACE ." J =" J . CR
      LOOP
   LOOP
;
```

LOOPS

LEAVING A DO ... LOOP OR DO ... +LOOP

THE ONLY WAY TO LEAVE ONE OF THE ABOVE LOOPS IS TO REACH THE LIMIT VALUE.

THE **FORTH WORD LEAVE**, WHEN EXECUTED BETWEEN DO AND LOOP OR +LOOP CHANGES THE STORED LIMIT VALUE TO THE CURRENT VALUE OF THE LOOP INDEX.
IN THIS CASE THE LOOP IS LEFT WHEN LOOP OR +LOOP IS REACHED NEXT TIME.

EXAMPLE:

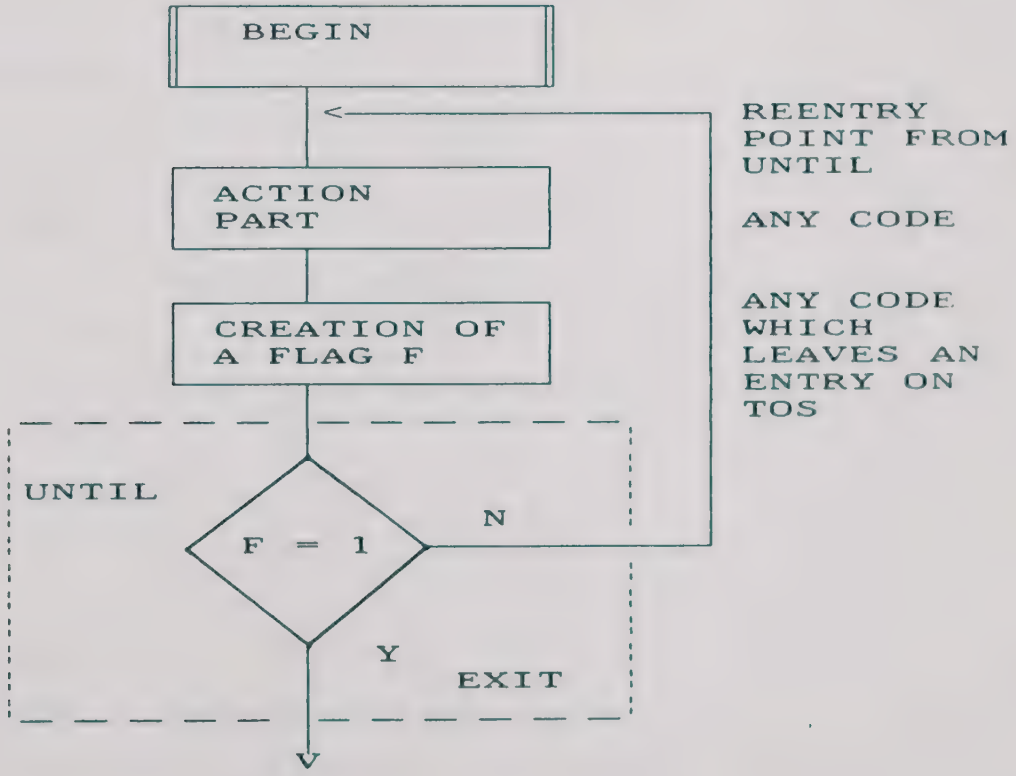
```
: TEST5
   10 0
   DO
      I DUP . 5 >
      IF
        LEAVE
      THEN
   LOOP
;
```

USE:

```
TEST5   CR

0 1 2 3 4 5 6 OK
```

LOOPS WITH UNDEFINED REPETITION NUMBER

BEGIN ... UNTIL



LOOPS WITH UNDEFINED REPETITION NUMBER

BEGIN ... UNTIL

EXAMPLE:

```
: TEST6 ( - )
   BEGIN
      ." PRESS Y KEY: "( ACTION PART )
      KEY
      89 =                ( CREATION OF FLAG)
   UNTIL
   ." END"
;
```

NOTES:

- UNTIL CONSUMES THE FLAG
- POSSIBLE CONTROL STRUCTURES IN THE ACTION PART MUST BE CLOSED

LOOPS WITH UNDEFINED REPETITION NUMBER

BEGIN ... AGAIN

THIS IS AN INFINITIE LOOP.

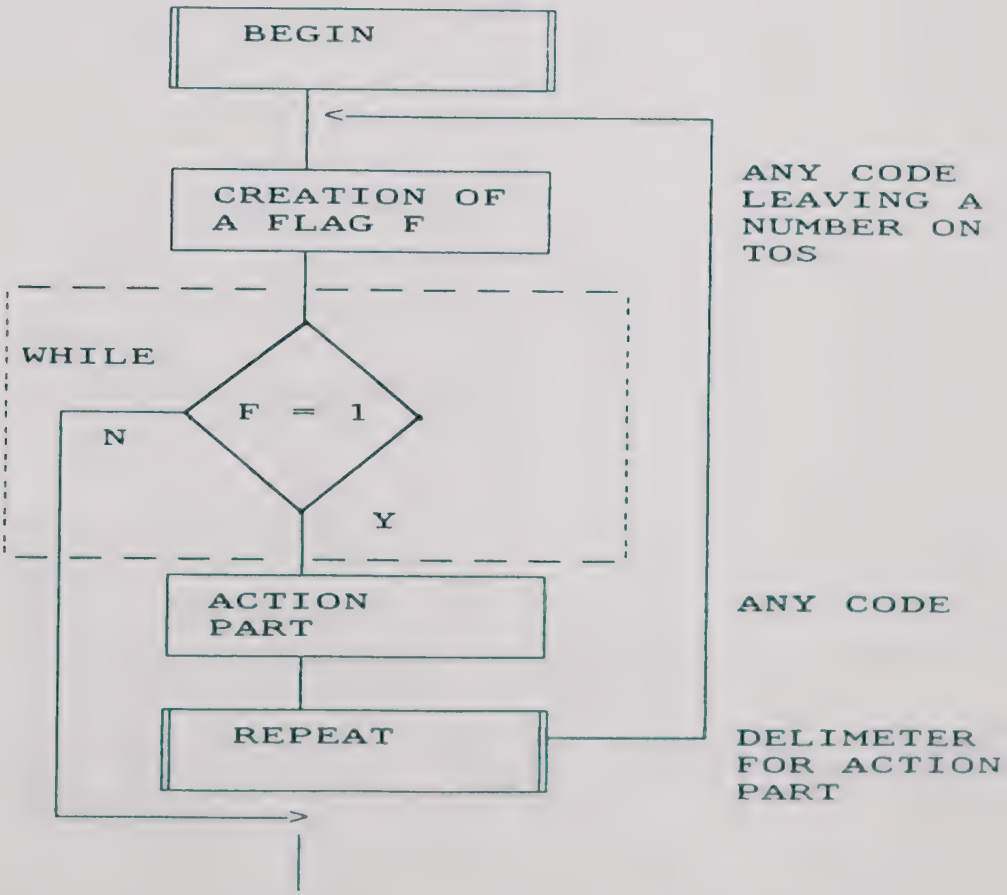
AGAIN CAUSES AN UNCONDITIONAL JUMP TO THE CODE AFTER **BEGIN**.

THERE IS NO SPECIAL FORTH WORD TO LEAVE SUCH A LOOP, BUT **EXIT** WHEN EXECUTED BETWEEN **BEGIN** AND **AGAIN** MAY SERVE FOR THIS PURPOSE.

"EXIT" LEAVES THE DEFINITION IN WHICH IT IS CALLED AND RETURNS TO THE DEFINITION WHICH CALLED IT OR TO THE OUTER INTERPRETER.

LOOPS WITH UNDEFINED REPETITION NUMBER

BEGIN ... WHILE ... REPEAT



LOOPS WITH UNDEFINED REPETITION NUMBER

BEGIN ... WHILE ... REPEAT

EXAMPLE:

```
: TEST7 ( - )
   BEGIN
      ." PRESS Y KEY: "
      KEY
      89 -                ( CREATION OF FLAG )
   WHILE
      ." WRONG CHOICE" CR ( ACTION PART )
   REPEAT
   ." RIGHT CHOICE" CR
;
```

NOTES:

- WHILE CONSUMES THE FLAG
- POSSIBLE CONTROL STRUCTURES IN THE ACTION PART MUST BE CLOSED
- UNLIKE DO ... LOOP OR BEGIN ... UNTIL THE TERMINATION CONDITION IS TESTED BEFORE THE ACTION PART

CASE CONSTRUCT

THE CASE CONSTRUCT PROVIDES A MECHANISM FOR SELECTING ONE OF A SET OF POSSIBLE CASES.

THE FOLLOWING EXAMPLE SHOWS THE USAGE OF THE RELATED WORDS:

```
( VALUE -- )
: TEST_CASE
   DOCASE
      CASE 1  { ." VALUE = 1" }
      CASE 2  { ." VALUE = 2" }
      CASE 3
      ORCASE 4  { ." VALUE = 3 OR 4" }
      CASE DUP  { ." OTHER VALUE" }
   ENDCASE
;
```

CASE CONSTRUCT

NOTES:

- IF EQUALITY IS FOUND, THE { }- ACTION IS PERFORMED AND THE PROGRAM EXECUTION CONTINUES AFTER ENDCASE; NO FURTHER COMPARISONS ARE EXECUTED

- CASE DUP EXPRESSES THE DEFAULT CONDITION AND MUST BE PLACED AS LAST CASE

WORDLIST

WORD	STACK NOTATION	MODULE
!	(D\ADDR --) STORES 32 BITS OF D AT ADDR	DA
#>>	SHIFTS 32 BIT VALUE BY N BITS TO THE RIGHT USE: N #>>	
( )	COMMENT	
*	(N1\N2 -- PROD) MULTIPLICATION	FC
*/	(N1\N2\N3 -- N1*N2\N3) LEAVES QUOTIENT	
+	(N1\N2 -- SUM) ADDITION	
+!	(D\ADDR --) INCREASES CONTENTS OF ADDR BY D	DA
+LOOP	LIKE LOOP, BUT THE INDEX IS CHANGED BY THE VALUE ON TOS	CO
-	(N1\N2 -- DIFF) SUBTRACTION	

WORD	STACK NOTATION	MODULE
.	OUTPUTS TOS ENTRY AS VALUE ACCORDING TO BASE; REMOVES TOS ENTRY	FC
."	COMPILER WORD FOR STRING OUTPUT	DE
.S	OUTPUTS TOP 4 STACK ENTRIES DOES NOT REMOVE THE ENTRIES	FC
/	(N1\N2 -- QUOTIENT) DIVISION	
/MOD	(N1\N2 -- REMAINDER\ QUOTIENT)	
:	DEFINING WORD FOR A FORTH WORD	DE
;	TERMINATES DEFINITION OF A FORTH WORD	DE
<	(N1\N2 -- FLAG) TRUE IF N1 < N2	FC
<<#	SHIFTS 32 BIT VALUE BY N BITS TO THE LEFT USE: N <<#	
=	(N1\N2 -- FLAG) TRUE IF N1 = N2	
>	(N1\N2 -- FLAG) TRUE IF N1 > N2	

WORD	STACK NOTATION	MODULE
>R	(N --) MOVES VALUE FROM PARAMETER STACK TO RETURN STACK	
?DUP	(N1 -- N1\N1) IF N1<>0 (N1 -- N1) IF N1= 0	
@	(ADDR -- VALUE) FETCHES 32 BIT VALUE FROM ADDR	DA
0<	(N -- FLAG) TRUE IF N < 0	
0=	(N -- FLAG) TRUE IF N = 0	
2DROP	(N1\N2 --)	
2DUP	(N1\N2 -- N1\N2\N1\N2)	
3DROP	(N1\N2\N3 --)	
3DUP	(N1\N2\N3 -- N1\N2\N3\ N1\N2\N3)	
ABS	(N -- |N|) LEAVES ABSOLUTE VALUE	
AGAIN	TERMINATES REPETITIVE SEQUENCE	CO
ALLOT	(N --) RESERVES N BYTES OF DICTIONARY SPACE	DA

WORD	STACK NOTATION	MODULE
ALLOTL	(N --) RESERVES N LONG WORDS OF DICTIONARY SPACE	DA
ALLOTW	(N --) RESERVES N WORDS OF DICTIONARY SPACE	DA
BEGIN	STARTS REPETITIVE SEQUENCE	CO
C!	(C\ADDR --) STORES 8 BITS OF C AT ADDR	DA
C@	(ADDR -- VALUE) FETCHES 8 BIT VALUE FROM ADDR	DA
CASE	PROVIDES MECHANISM FOR SELECTING ONE OF A SET OF POSSIBLE CASES	CO
CFA	(PFA -- CFA)	DA
CMOVE	(ADDR1\ADDR2\N --) MOVES N BYTES BEGINNING AT ADDR1 TO ADDR2	DA
CONSTANT	DEFINING WORD FOR A CONSTANT	DA
COUNT	(ADDR -- ADDR+1\N)	DA
CR	NEXT OUTPUT IN NEW LINE	

WORD	STACK NOTATION	MODULE
DO	(LIMIT\INDEX --) STARTS DO LOOP	CO
DP	DICTIONARY POINTER	DA
DROP	(N --)	FC
DUP	(N1 -- N1\N1)	FC
ELSE	SEPERATES "TRUE PART" FROM "FALSE PART"	CO
EMIT	(C --)	DA
ENDIF	SYNONYM TO THEN	CO
EXECUTE	(CFA --) EXECUTES FORTH WORD WITH CODE FIELD ADDRESS CFA	DA
EXIT	LEAVE DEFINITION	
FORGET	EXECUTED IN THE FORM: **FORGET** CCCC DELETES DEFINITION NAME CCCC FROM THE DICTIONARY WITH ALL ENTRIES PHYSICALLY FOLLOWING IT (BEING ABOVE CCCC)	
HERE	SAME AS DP @	DA
I	(-- LOOP INDEX) INNERMOST LOOP	CO

WORD	STACK NOTATION	MODULE
IF	(FLAG --)	CO
J	(-- LOOP INDEX) NEXT OUTER LOOP	CO
KEY	(-- C)	DA
LEAVE	EARLY TERMINATON OF A LOOP	CO
LOOP	TERMINATES REPETITIVE SEQUENCE	CO
MAX	(N1\N2 -- MAXIMUM VALUE OF N1,N2)	
MIN	(N1\N2 -- MINIMUM VALUE OF N1,N2)	
MOD	(N1\N2 -- REMAINDER) OF N1/N2	
MOVE	(ADDR1\ADDR2\N --) MOVES N WORDS BEGINNING AT ADDR1 TO ADDR2	
OVER	(N1\N2 -- N1\N2\N1)	FC
PICK	(N -- M) COPY OF Nth STACK ENTRY ON TOS	FC
QUERY	EXPECTS 80 CHARACTERS OF TEXT (OR UNTIL CR) AND SAVE IT TO TIB	DA

WORD	STACK NOTATION	MODULE
R	(-- N) COPIES VALUE FROM RETURN STACK TO PARAMETER STACK	
R>	(-- N) MOVES VALUE FROM RETURN STACK TO PARAMETER STACK	
REPEAT	TERMINATES REPETITIVE SEQUENCE	CO
ROT	(N1\N2\N3 -- N2\N3\N1)	FC
S.	OUTPUTS **TOS** ENTRY AS VALUE ACCORDING TO **BASE**; DOES NOT REMOVE **TOS** ENTRY	FC
SPACE	NEXT OUTPUT SHIFTED BY ONE SPACE	
SPACES	(N --) NEXT OUTPUT SHIFTED BY N SPACES	DE
STACK	OUTPUTS **COMPLETE STACK** WITHOUT REMOVING THE VALUES	FC
STYPE	(CHAR ADDRESS\ CHAR COUNT --) OUTPUT A STRING	DA
STYPE_A	(CHAR ADDRESS\ CHAR COUNT\ATTR --) STYPE WITH ATTRIBUTE	DA

WORD	STACK NOTATION	MODULE
SWAP	(N1\N2 -- N2\N1)	FC
THEN	DELIMITING WORD FOR CONDITIONED EXECUTION	CO
UNTIL	(FLAG --) TERMINATES REPETITIVE SEQUENCE	CO
VARIABLE	DEFINING WORD FOR A VARIABLE	DA
VLIST	LISTS THE NAMES OF THE DEFINITIONS IN THE DICTIONARY	
W!	(N\ADDR --) STORES 16 BITS OF N AT ADDR	DA
W@	(ADDR -- VALUE) FETCHES 16 BIT VALUE FROM ADDR	DA
WHILE	(FLAG --) KEYWORD FOR CONDITIONALLY REPETITION	CO
WORD	(C --) READS THE NEXT CHAR. FROM INPUT STREAM UNTIL DELIMETER WITH ASCII VALUE C IS FOUND	DA

EXERCISE 1

IF ... ELSE ... THEN

A) DEFINE A WORD "EQUAL" WHICH TAKES TWO NUMBERS FROM THE STACK AND GIVES A MESSAGE IF THE NUMBERS ARE EQUAL.

B) DEFINE A WORD "SIGN" WHICH EXPECTS A NUMBER ON TOS AND GIVES A MESSAGE WHETHER IT IS POSITIVE, NEGATIVE OR ZERO .

C) DEFINE A WORD "CHECK" WHICH EXPECTS A NUMBER ON TOS AND PRINTS

TRUE -- IF THE NUMBER IS POSITIVE AND LESS THAN 15

TRUE -- IF THE NUMBER IS NEGATIVE AND GREATER THAN -5

TRUE -- IF THE NUMBER IS EQUAL -27

FALSE -- IN ANY OTHER CASE

NOTE: "CHECK" SHOULD LEAVE THE NUMBER ON TOS UNCHANGED.

EXERCISE 2

DO ... LOOP

A) DEFINE A WORD "7ER" WHICH DISPLAYS THE MULTIPLES OF 7 BETWEEN 1 AND 70.

B) DEFINE A WORD "8ER" BUT DO NOT USE THE WORD * THIS TIME. USE THE WORD +LOOP INSTEAD.

C) DEFINE A WORD "MUL" WHICH EXPECTS 2 NUMBERS M AND N ON THE STACK.
MUL SHOULD CALCULATE AND DISPLAY THE MULTIPLES OF M FROM 1 TO 100.
IF THE NUMBER CALCULATED GETS GREATER THAN N, MUL SHOULD STOP.

D) DEFINE A WORD "MULTAB" WHICH DISPLAYS A MULTIPLICATION TABLE IN THE FORM

```
 1   2   3   ...   10
 2   4   6   ...   20
 .                  .
 .                  .
 .                  .
10  20  30   ...  100
```

HINTS:
- USE A NESTED DO ... LOOP CONSTRUCTION TO CALCULATE THE VALUES
- USE 5 .R TO DISPLAY THE VALUES RIGHT ALIGNED IN A 5 CHARACTER WIDE FIELD EACH.

EXERCISE 3

BEGIN ... UNTIL

A) DEFINE A WORD "WAIT" WHICH RUNS IN A LOOP UNTIL THE CHARACTER "A" HAS BEEN PRESSED 10 TIMES AT THE KEYBOARD. A MESSAGE SHOULD BE DISPLAYED WHEN THE LOOP IS LEFT.

HINT: THE DECIMAL VALUE OF THE ASCII CHARACTER "A" IS 65.

B) DEFINE A WORD "FILBUF" WHICH EXPECTS THE 3 VALUES BUF, N AND CH ON THE STACK AND WHICH FILLS N BYTES STARTING AT THE ADDRESS BUF WITH THE CHARACTER CH.

HINT: USE 3 VARIABLES BUF, N AND CH FOR THIS EXERCISE.

FOR EXPERTS:
REPEAT THE EXERCISE, BUT DO NOT USE VARIABLES THIS TIME. MAKE USE OF THE RETURN STACK AS A TEMPORARY VARIABLE.

EXERCISE 3

C) DEFINE A WORD "NWORD" WHICH EXPECTS AN ADDRESS OF A TEXTBUFFER AND RETURNS THE ADDRESS OF THE FIRST BYTE OF THE FOLLOWING WORD IN THAT BUFFER. THE WORD SEPARATOR IS ONE OR MULTIPLE SPACES.

EXAMPLE: THIS IS A TEST TEXT.
GIVEN THE ADDRESS OF THE FIRST "T" OF "THIS" NWORD RETURNS THE ADDRESS OF THE "I" IN "IS" ETC.

HINTS:
- USE THE VARIABLE "PTR" TO STORE THE CURRENT ADDRESS.
- MAKE USE OF THE WORD <> (NOT EQUAL)

```
: <>    (N\M --- F)
  = 0=
;
```

- THE DECIMAL VALUE OF THE ASCII CHARACTER "SPACE" IS 32.

EXERCISE 4

BEGIN ... WHILE ... REPEAT

A) DEFINE A WORD "KCOUNT" WHICH COUNTS THE NUMBER OF KEYSTROKES UNTIL AN "A" IS ENTERED. KCOUNT SHOULD DISPLAY THE CURRENT NUMBER OF KEYSTROKES EACH TIME A KEY IS PRESSED AND GIVE AN "END" MESSAGE WHEN THE "A" IS ENTERED.

B) REWRITE THE WORD "FILBUF" (CO_X-3B) USING THE BEGIN ... WHILE ... REPEAT CONSTRUCTION.

C) REWRITE THE WORD "NWORD" (CO_X-3C) USING THE BEGIN ... WHILE ... REPEAT CONSTRUCTION.

EXERCISE 5

CASE

A) DEFINE A WORD "NDIGIT" WHICH EXPECTS A SINGLE DIGIT ON THE STACK AND DISPLAYS THE NAME OF THAT DIGIT.
EXAMPLE: 3 NDIGIT DISPLAYS "THREE"
9 NDIGIT DISPLAYS "NINE"

B) IMPROVE YOUR WORD "NDIGIT" BY ADDING AN EXTRA CASE WHICH GIVES AN ERROR MESSAGE IF THE NUMBER ON THE STACK IS NOT A SINGLE DIGIT OR IF IT IS NEGATIVE.

C) DEFINE A WORD "CHARCOUNT" WHICH COUNTS HOW OFTEN CERTAIN KEYS ARE PRESSED. "CHARCOUNT" SHOULD USE A NUMBER OF VARIABLES (e.g. A, B, C, D ...) TO COUNT THE CHARACTERS. UNEXPECTED CHARACTERS SHOULD BE COUNTED IN THE VARIABLE "OTHER" .
THE CURRENT VALUES OF ALL COUNTER VARIABLES SHOULD BE DISPLAYED AFTER EVERY KEYSTROKE (DEFINE THE WORD "C_DISPLAY" FOR THIS PURPOSE.
THE PROGRAM SHOULD TERMINATE IF A "CR" (CARRIAGE RETURN) IS ENTERED.

SOLUTION 1

IF .. ELSE .. THEN

A)
```
: EQUAL                          ( N\M -- )

  =
  IF ." THE NUMBERS ARE EQUAL"  CR
  THEN
;
```

B)
```
: SIGN                           ( N --   )

  DUP
  0<                      ( PRODUCE A FLAG )
  IF    ." THE NUMBER IS NEGATIVE" CR DROP
  ELSE 0=                 ( PRODUCE A FLAG )
       IF    ." THE NUMBER IS ZERO"       CR
       ELSE ." THE NUMBER IS POSITIVE"   CR
       THEN
  THEN
;
```

C)
```
: CHECK                          ( N -- N )

  DUP
  -27 =
  IF                  ." TRUE"  CR
  ELSE DUP DUP 0< SWAP -5 > AND
       IF             ." TRUE"  CR
       ELSE DUP DUP 0 > SWAP 15 < AND
            IF        ." TRUE"  CR
            ELSE      ." FALSE" CR
            THEN
       THEN
  THEN
;
```

SOLUTION 2

DO ... LOOP

A)
```
: 7ER                          ( -- )

  11 1
  DO
    I 7 * . CR
  LOOP
;
```

B)
```
: 8ER                          ( -- )

  81 8
  DO
    I . CR 8
  +LOOP
;
```

SOLUTION 2

C)

```
: MUL                       ( M\N --  )

  SWAP                      ( N\M     )
  101 1
  DO
    2DUP                  ( N\M\N\M   )
    I *                   ( N\M\N\M*I )
    DUP .
    <                       ( N\M\F   )
    IF   LEAVE
    THEN
  LOOP                      ( N\M     )
  2DROP                     ( --      )
;
```

D)

```
: MULTAB                    (  --     )

  11 1
  DO                    ( OUTER LOOP  )
    11 1
    DO                  ( INNER LOOP  )
        I J *           ( CALCULATION )
        5 .R            ( DISPLAY     )
    LOOP                ( NEXT I      )
    CR                  ( NEW LINE    )
  LOOP                  ( NEXT J      )
;
```

SOLUTION 3

BEGIN ... UNTIL

A)

```
0  VARIABLE  COUNTER


: WAIT                          ( -- )

  0  COUNTER  !
  BEGIN
    KEY 65 =
    IF
        1  COUNTER +!
    THEN
    COUNTER @  10 =
  UNTIL
  CR ." PROGRAM END"
;
```

B)

```
0 VARIABLE BUF
0 VARIABLE N
0 VARIABLE CH

: FILBUF              ( BUF\N\CH --                  )

  CH C! N ! BUF !     ( STORE THE VALUES             )
  BEGIN
    CH C@ BUF @ C!    ( STORE CH AT BUF              )
     1 BUF +!         ( POINT TO NEXT LOCATION       )
    -1 N   +!         ( DECREMENT NO. OF BYTES       )
    N @ 0=            ( PRODUCE A FLAG               )
  UNTIL               ( REPEAT IF NOT TRUE           )
;
```

SOLUTION 3

FOR EXPERTS:

```
: FILBUF                                  ( BUF\N\CH --  )

                  PARAMETER STACK          RETURN STACK

  ROT            ( N\CH\BUF        )  (     --       )
  BEGIN          ( N\CH\BUF        )  (     --       )
    2DUP C!      ( N\CH\BUF        )  (     --       )
    ROT 1 -      ( CH\BUF\N-1      )  (     --       )
    DUP >R       ( CH\BUF\N-1      )  (     N-1      )
    ROT ROT 1+   ( N-1\CH\B+1      )  (     N-1      )
    R>           ( N-1\CH\B+1\N-1  )  (     --       )
    0=           ( N-1\CH\B+1\F    )  (     --       )
  UNTIL          ( N-1\CH\B+1      )  (     --       )
  2DROP DROP     (      ---        )  (     --       )
;
```

SOLUTION 3

C)

```
: <>                                ( N\M -- F      )
  = 0=
;

0 VARIABLE PTR

: FIND_SPACE                        (     --        )

  BEGIN
    PTR @ C@                 ( GET CHARACTER VALUE      )
    1 PTR +!                 ( POINT TO NEXT CHARACTER  )
    32 =                     ( IS IT A SPACE ?          )
  UNTIL                      ( REPEAT UNTIL TRUE        )
  -1 PTR +!                  ( RETURN POINTER TO THIS   )
;                            ( SPACE                    )

: SKIP_SPACES                       (     --        )

  BEGIN
    1 PTR +!                 ( POINT TO NEXT CHARACTER  )
    PTR @ C@                 ( GET CHARACTER VALUE      )
    32 <>                    ( IS IT A NON SPACE ?      )
  UNTIL                      ( REPEAT UNTIL TRUE        )
;

: NWORD                             ( BUF --  PTR   )

  PTR !                    ( POINT TO BUFFER ADDRESS  )
  FIND_SPACE               ( POINT TO NEXT SPACE      )
  SKIP_SPACES              ( POINT TO NEXT NON SPACE  )
  PTR @                    ( RETURN POINTER ON STACK  )
;
```

SOLUTION 4

BEGIN ... WHILE ... REPEAT

A)

```
0 VARIABLE CNT

: KCOUNT                                        ( --  )

  0 CNT !              ( RESET COUNTER                 )
  BEGIN
    KEY                ( WAIT FOR A KEYSTROKE          )
    65 <>              ( SOMETHING ELSE THAN "A" ?     )
  WHILE
    1 CNT +!           ( INCREMENT COUNTER             )
    CNT @ CR .         ( DISPLAY CURRENT COUNT         )
  REPEAT
  CR
  ." PROGRAM END"      ( DISPLAY "END" MESSAGE         )
;
```

SOLUTION 4

B)

```
0 VARIABLE BUF
0 VARIABLE N
0 VARIABLE CH

: FILBUF                              ( BUF\N\CH --  )

  CH C! N ! BUF !     ( STORE THE VALUES          )
  BEGIN
    N @ 0 >           ( STILL BYTES TO FILL ?     )
  WHILE
    CH C@ BUF @ C!    ( FILL THE CURRENT BYTE     )
     1 BUF +!         ( POINT TO THE NEXT BYTE    )
    -1 N   +!         ( DECREMENT NO. OF BYTES    )
                      ( TO BE FILLED              )
  REPEAT              ( DO THE LOOP AGAIN         )
;
```

SOLUTION 4

C)

```
0 VARIABLE PTR

: FIND_SPACE                                    ( -- )

  BEGIN
    PTR @ C@ ( GET CHARACTER VALUE                    )
    32 <>    ( IS IT ANYTHING ELSE THAN "SPACE" ? )
  WHILE      ( WHILE IT IS                            )
    1 PTR +! ( POINT TO THE NEXT CHARACTER            )
  REPEAT     ( DO THE LOOP AGAIN                      )
;

: SKIP_SPACES                                   ( -- )

  BEGIN
    PTR @ C@ ( GET CHARACTER VALUE                    )
    32 =     ( IS IT A "SPACE" ?                      )
  WHILE      ( WHILE IT IS                            )
    1 PTR +! ( POINT TO THE NEXT CHARACTER            )
  REPEAT     ( DO THE LOOP AGAIN                      )
;

: NWORD                                     ( BUF -- PTR )

  PTR !        ( POINT TO THE BUFFER ADDRESSS         )
  FIND_SPACE   ( POINT TO THE NEXT "SPACE"            )
  SKIP_SPACES  ( POINT TO THE NEXT "NON SPACE"        )
  PTR @        ( RETURN POINTER ON STACK              )
;
```

SOLUTION 5

CASE

A)

```
: NDIGIT                                        ( N --  )
                CR
                DOCASE
                  CASE 0  { ." ZERO " }
                  CASE 1  { ." ONE  " }
                  CASE 2  { ." TWO  " }
                  CASE 3  { ." THREE" }
                  CASE 4  { ." FOUR " }
                  CASE 5  { ." FIVE " }
                  CASE 6  { ." SIX  " }
                  CASE 7  { ." SEVEN" }
                  CASE 8  { ." EIGHT" }
                  CASE 9  { ." NINE " }
                ENDCASE
              ;
```

B)

```
: NDIGIT                                        ( N --  )
        CR
        DOCASE
          CASE 0    { ." ZERO " }
          CASE 1    { ." ONE  " }
          CASE 2    { ." TWO  " }
          CASE 3    { ." THREE" }
          CASE 4    { ." FOUR " }
          CASE 5    { ." FIVE " }
          CASE 6    { ." SIX  " }
          CASE 7    { ." SEVEN" }
          CASE 8    { ." EIGHT" }
          CASE 9    { ." NINE " }
          CASE DUP  ( ." NOT A SINGLE CHARACTER OR NEGATIVE" }
        ENDCASE
      ;
```

SOLUTION 5

```
C )    0 VARIABLE A
       0 VARIABLE B
       0 VARIABLE C
       0 VARIABLE OTHER


       : C_DISPLAY                        ( -- )

           CR
           ." A= "     A      @ . 5 SPACES
           ." B= "     B      @ . 5 SPACES
           ." C= "     C      @ . 5 SPACES
           ." OTHER=" OTHER @ .

         ;


       : CHARCOUNT                        ( -- )

         BEGIN
           KEY  DUP                       ( GET KEYBOARD INPUT     )
           DOCASE
             CASE 65   { 1 A +! }         ( "A" HAS BEEN ENTERED   )
             CASE 66   { 1 B +! }         ( "B" HAS BEEN ENTERED   )
             CASE 67   { 1 C +! }         ( "C" HAS BEEN ENTERED   )
             CASE DUP { 1 OTHER +! }      ( ANYTHING ELSE HAS      )
                                          ( BEEN ENTERED           )
           ENDCASE
           C_DISPLAY                      ( DISPLAY VALUES         )
           13 =                           ( FLAG FOR UNTIL         )
         UNTIL                            ( REPEAT FROM "BEGIN"    )
;
```

LAPD INTRODUCTION

SUMMARY

- LAPD MONITOR/ANALYZER
- LAPD EMULATION
- LAPD TESTMANAGER

LAPD MONITOR/ANALYZER

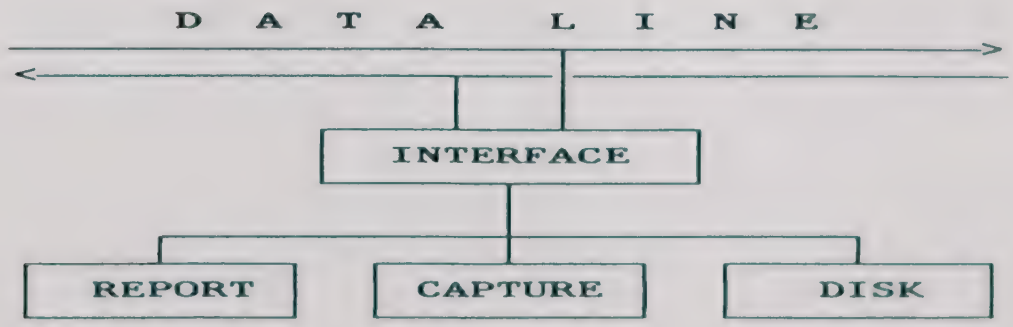
SUMMARY

- DATA CHANNELS AND DATA FLOW
- MONITOR STATES
- ANALYZER FEATURES

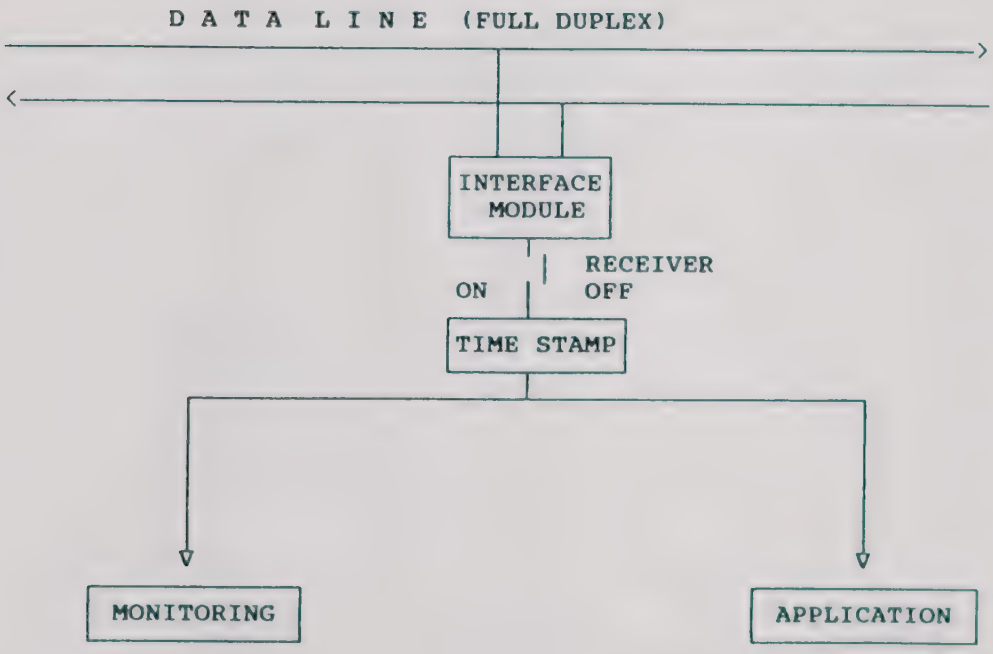
DATA CHANNELS AND DATA FLOW

ONE OF THE BASIC APPLICATION FUNCTIONS OF THE K1197 IS DATA MONITORING. THE INCOMING DATA STREAM IS INTERNALLY SPLIT INTO THREE INDEPENDENT DATA CHANNELS:

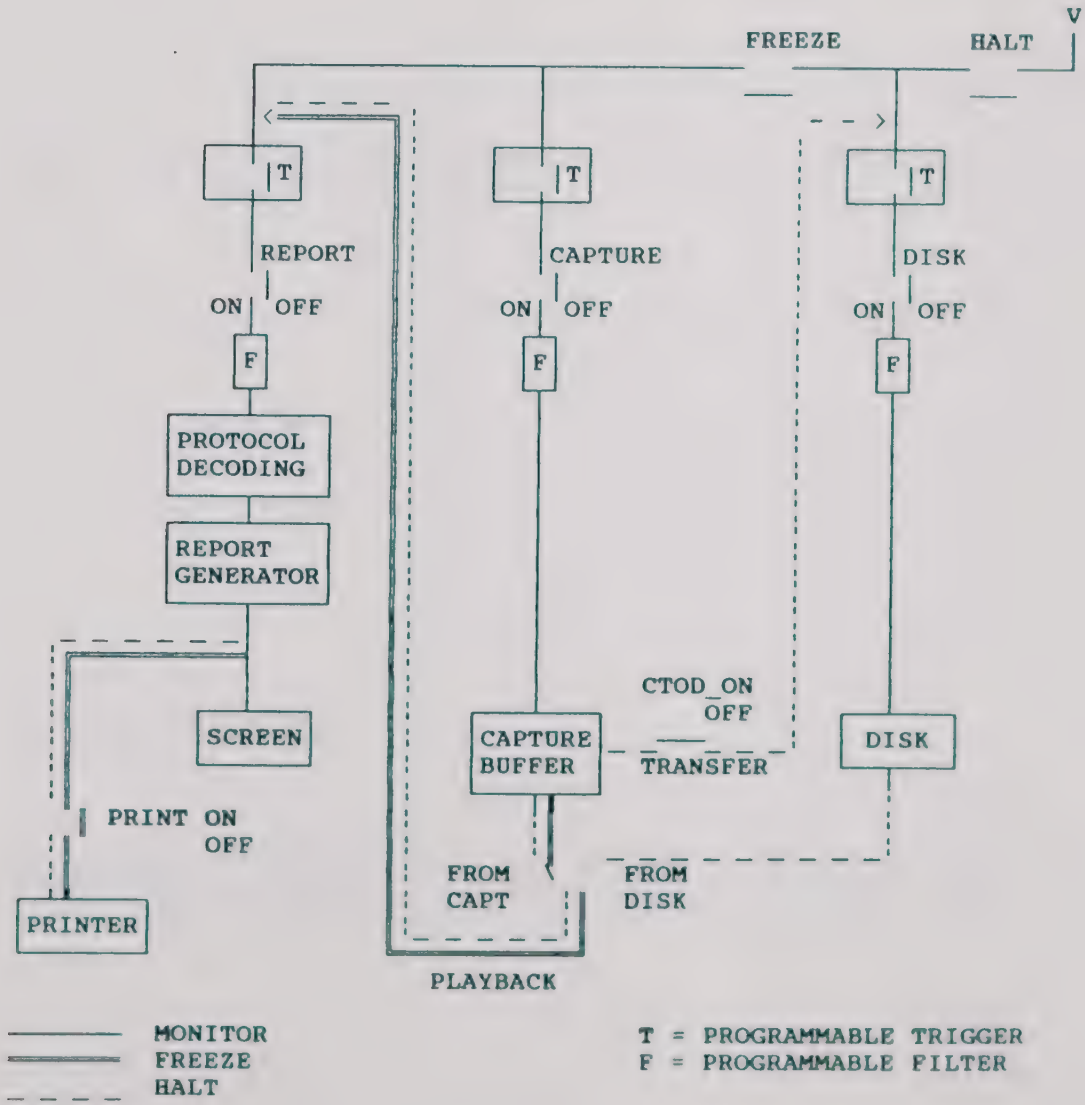
- TO DISPLAY (REPORT FUNCTION)
- TO CAPTURE BUFFER (CAPTURE FUNCTION)
- TO DISK (RECORD FUNCTION)



DATA CHANNELS AND DATA FLOW



DATA CHANNELS AND DATA FLOW



MONITOR STATES

THE MONITORING PROGRAM IS ALWAYS IN ONE OF THE THREE STATES:

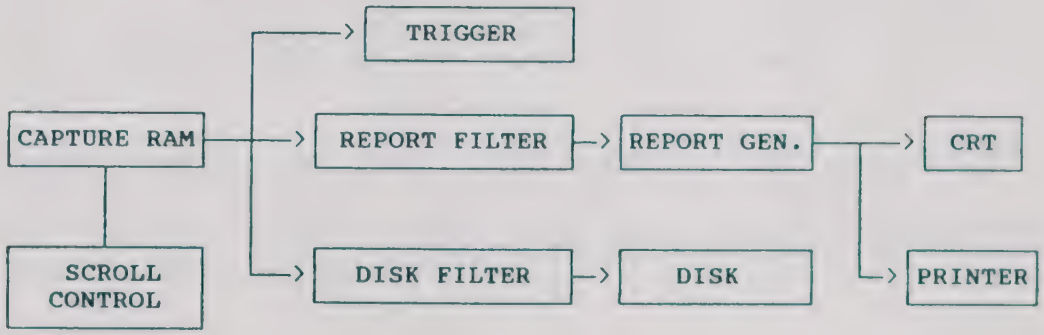
- HALT
- FREEZE
- MONITOR

HALT STATE

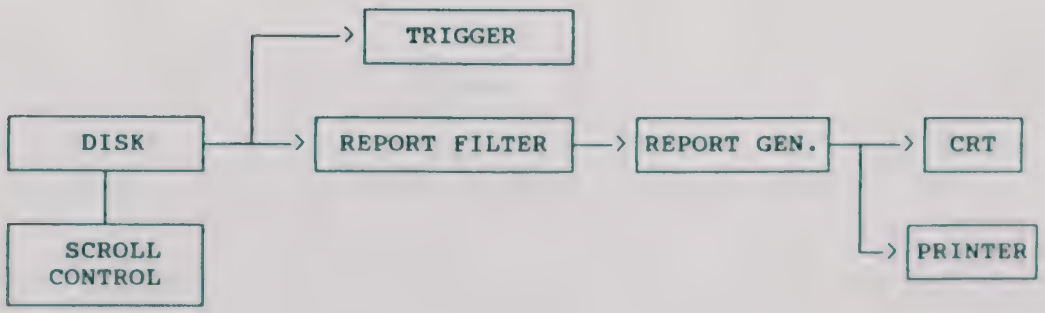
ALL PROCESSING OF LIVE DATA IS HALTED. SCROLLING AND DISPLAY CAN BE DONE FROM CAPTURE RAM OR FROM DISK PLAYBACK.

MONITOR STATES

HALT STATE



DATA FROM CAPTURE RAM



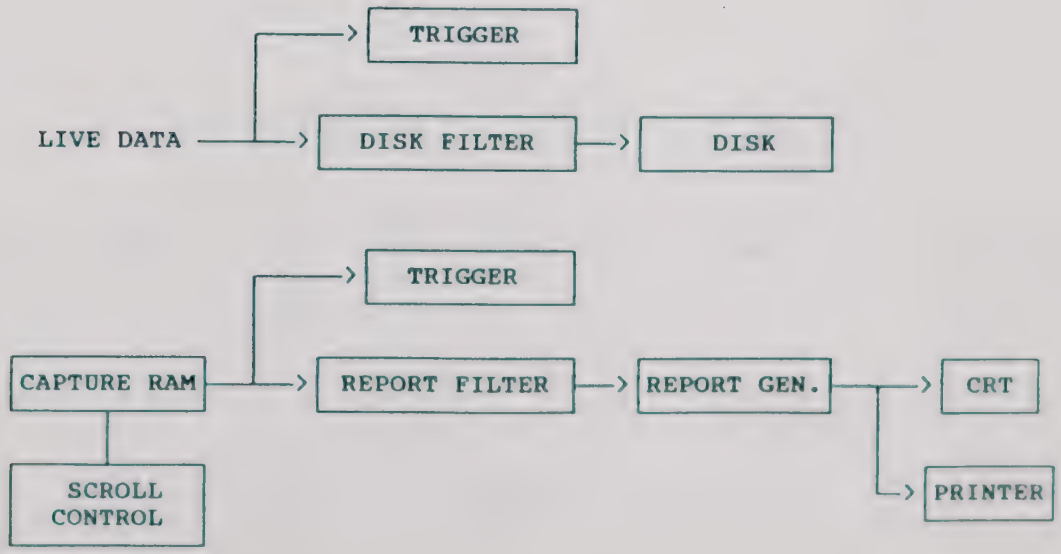
DATA FROM DISK PLAYBACK

MONITOR STATES

FREEZE STATE

THE DISPLAY AND CAPTURE BUFFER CAN BE FROZEN AT ANY TIME FOR AN INSTANT REPLAY OF THE CAPTURED DATA WHILE DATA RECORDING TO DISK MAY STILL BE ACTIVE.

LIVE DATA IS NOT ALLOWED TO GO TO THE DISPLAY REPORT OR TO THE CAPTURE RAM. HOWEVER, LIVE DATA CAN STILL BE RECORDED ON DISK. THE CAPTURE RAM CAN BE SCROLLED TO THE DISPLAY.

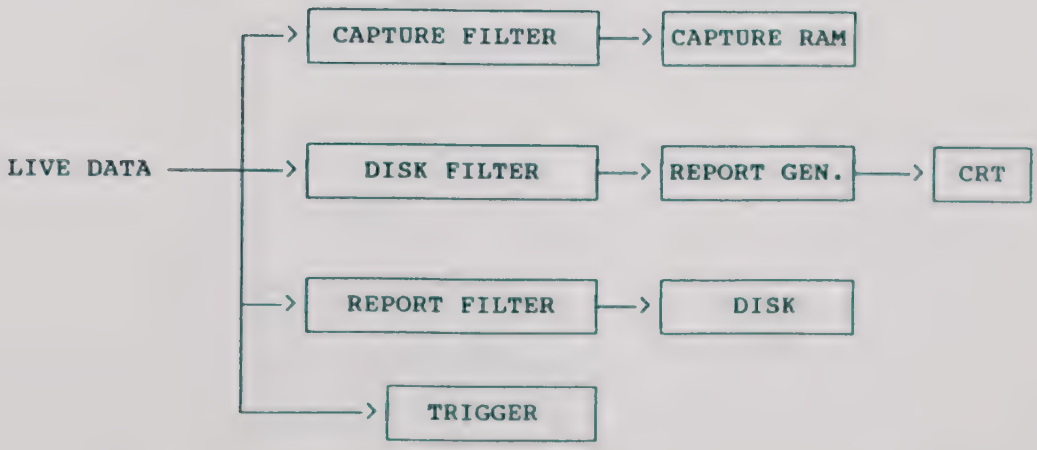


MONITOR STATES

MONITOR STATE

LIVE DATA IS BEING MONITORED AND WILL BE TAKEN AS SOURCE FOR DISPLAY REPORT, CAPTURE RAM AND DISK RECORDING.

IN THE MONITOR STATE THE USER MAY REPORT DATA, STORE IT IN THE CAPTURE RAM AND RECORD IT TO DISK AT THE SAME TIME.



ANALYZER FEATURES

- MONITORING OF LIVE DATA AND CONTROL LEAD CHANGES
- REPORT OF ALL DATA IN SEVERAL FORMATS ON SCREEN OR PRINTER
- CAPTURING OF DATA IN CAPTURE RAM
- RECORDING OF DATA ON DISK
  - LIVE DATA
  - CAPTURED DATA
- PLAYBACK OF DATA FROM DISK
  - DISPLAY
  - PRINTOUT
- FILTER FUNCTIONS
- TRIGGER FUNCTIONS

ANALYZER FEATURES

PROTOCOL DECODING

- THIS MODULE DECODES THE DATA RECEIVED FROM THE DATA LINE OR PLAYBACK MEDIA

HARD COPY PRINTOUT

- ANY INFORMATION DISPLAYED ON THE SCREEN CAN BE PRINTED TO A SERIAL ASYCHRONOUS PRINTER CONNECTED TO THE AUXILIARY RS232-C/V.28 PRINTER PORT

REPORT GENERATOR

- PROCESSING OF DATA FOR THE DISPLAY
- PROVIDES DIFFERENT OPTIONS AND DISPLAY MODES

ANALYZER FEATURES

DATA CAPTURING

- ALL RECEIVED DATA BLOCKS AND TRACE REPORTS ARE TIME STAMPED WITH 0.1 msec RESOLUTION.

- THE CAPTURE RATE AMOUNTS UP TO 72 Kbits/sec FULL DUPLEX FOR SYNCHRONOUS AND 19.2 Kbits/sec FOR ASYNCHRONOUS DATA STREAMS.

- 64 Kbytes OF RAM CAPTURE BUFFER AND 2x 790 Kbytes (2x 1.3 Mbytes for HD) OF DISK CAPTURE BUFFER. UP TO 40 Mbytes OF HARD DISK CAPTURE BUFFER.

- CONTROL IS CARRIED OUT BY START/STOP FUNCTION KEY, WRAPAROUND (ENDLESS LOOP) OR STOP WHEN FULL AND TRIGGER CONTROLLED START/STOP.

ANALYZER FEATURES

DATA RECORDING

- DATA RECORDING TO FLOPPY DISK (DD/HD = 790 Kbytes/1.3 Mbytes formated capacity) OR TO 40 Mbyte HARDDISK POSSIBLE.

- THE FILE SIZE IS USER SELECTABLE FROM 10 Kbytes TO MAXIMUM CAPACITY IN STEPS OF 10 DISK BLOCKS (≈5 KBYTES).

- ALL CAPTURED INFORMATION INCLUDING LINE DATA, TIME STAMPS AND TRACE REPORTS CAN BE RECORDED TO DISK IN REAL-TIME.

- RECORDING IS CONTROLLED BY START/STOP FUNCTION KEY AND TRIGGER, WRAPAROUND (ENDLESS LOOP) OR STOP WHEN DISK FILE FULL.

ANALYZER FEATURES

DATA PLAYBACK

- THE PLAYBACK FUNCTION ALLOWS FOR VIEWING AND PRINTING OF DATA FROM THE CAPTURE BUFFER OR THE DISK
- PLAYBACK IS POSSIBLE LINE BY LINE OR PAGE BY PAGE

DATA TRANSFER

- TRANSFER FROM CAPTURE BUFFER TO DISK
- SELECTION (BY FILTERS) OF SPECIFIC DATA BLOCKS FOR TRANSFER

LAPD EMULATION INTRODUCTION

SUMMARY

- FEATURES
- MENU SYSTEM
- SCREEN LAYOUT
- GENERAL AND REPORT COMMANDS
- DATA LINK SELECTION COMMANDS
- SERVICE PRIMITIVES
- FIRST SEND COMMANDS

FEATURES

REFERENCE EMULATION FEATURES

- EMULATION OF TERMINAL (TE) AND NETWORK (NET)
- HANDLING OF 8 DATA LINKS (AS NET) OR SIMULATING 8 TERMINAL EQUIPMENTS (TE), IDENTIFIED BY A CONNECTION ENDPOINT SUFFIX (CES) BETWEEN 0 AND 7
- AUTOMATIC AND MANUAL TEI ASSIGNMENT FOR EACH CES
- FREE SAPI SELECTION FOR EACH CES
- AUTOMATIC CORRECT PROTOCOL RESPONSES TO ALL RECEIVED FRAMES
- USER COMMANDS TO SEND CORRECT OR INCORRECT FRAMES IN CONFORMANCE TO THE PROTOCOL OR NOT
- EXECUTION OF ALL DISPLAY AND ANALYZER FUNCTIONS OF THE LAPD ANALYZER
- EXECUTION OF USER DEFINED SCENARIOS BY USING THE TESTMANAGER

MENU SYSTEM

- CURSOR-DRIVEN POPUP MENUS WITH ITEM SELECTION BY <<CR>> KEY
- FUNCTION KEYS FOR <<GENERAL>> FEATURES
- ITEM SELECTION BY USER COMMANDS POSSIBLE

TYPICAL FUNCTION KEY FEATURES:

1 QUIT_MENU 2 HELP 3 COMMAND 4 5 6 7 8

QUIT_MENU: LEAVE POPUP MENU AND SHOW DATA WINDOW

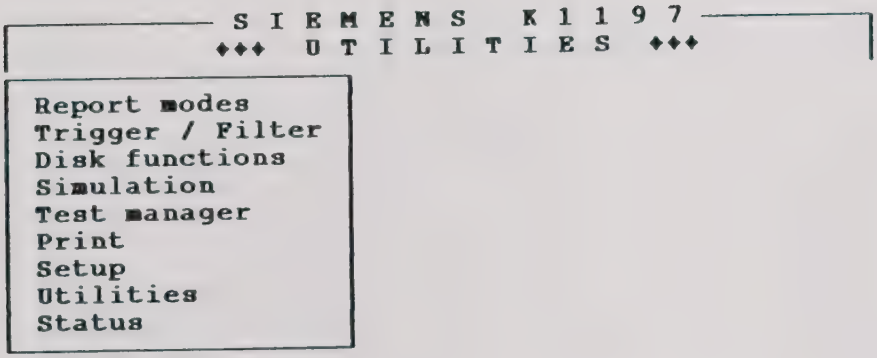
HELP: DISPLAY HELP INFORMATION

COMMAND: SWITCH TO COMMAND MODUS TO ENTER COMMANDS FROM THE KEYBOARD

OTHER FUNCTION KEYS MAY BE ACTIVE IN PARTICULAR POPUP MENUS

MENU SYSTEM

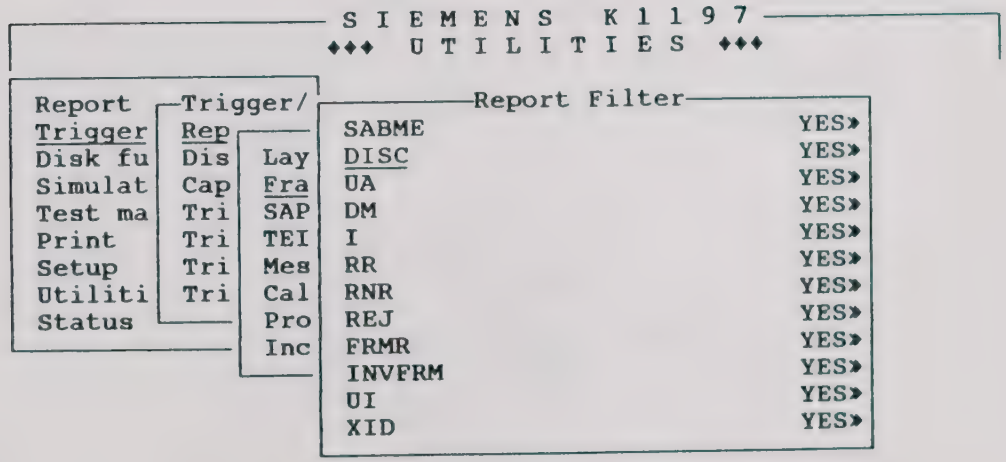
ROOT MENU PAGE



- ROOT ITEM SELECTION BY VERTICAL CURSOR MOVEMENT
- ITEM REFINEMENT: MOVE CURSOR RIGHT TO OPEN NEW POPUP MENU
- REPEAT SELECTION UNTIL FINAL SUBITEM IS REACHED. SELECT BY <<CR>> KEY

MENU SYSTEM

EXAMPLE: REPORT FILTER SELECTION

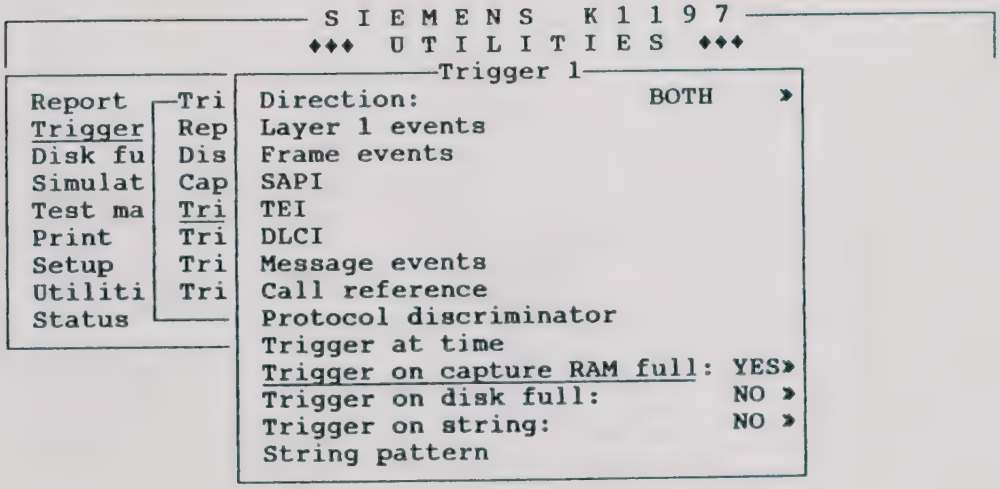


TREE: TRIGGER/FILTER -> REPORT FILTER
-> FRAME EVENTS

ALL/NONE SELECTION POSSIBLE IN
THIS MENU BY FUNCTION KEYS

MENU SYSTEM

EXAMPLE: SET TRIGGER ON CAPTURE RAM FULL

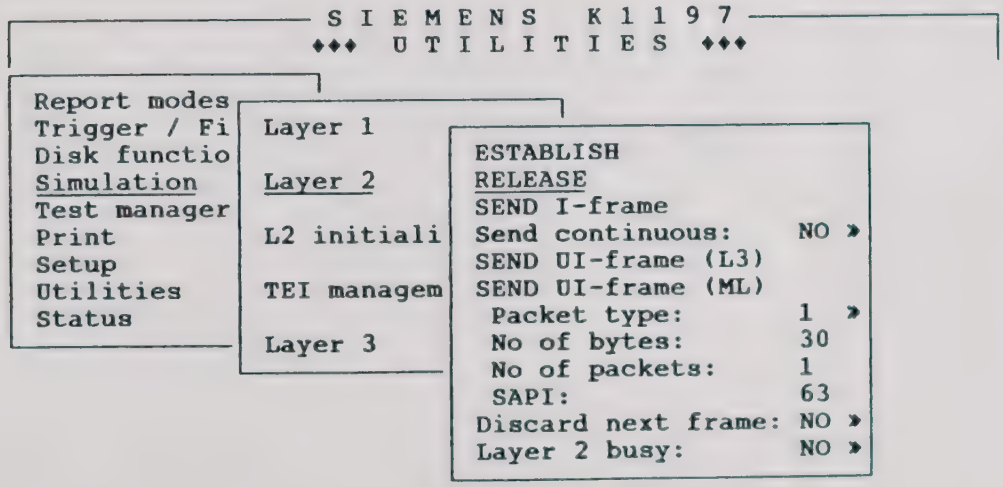



```
ISDN D-CHANNEL EMULATION/SIMULATION Version 2.0  Rev. 0 alpha
OK-7
1 QUIT_MENU 2 HELP 3 COMMAND  4       5       6       7       8
```

TREE: TRIGGER/FILTER -> TRIGGER 1
-> TRIGGER ON CAPTURE RAM FULL

MENU SYSTEM

EXAMPLE: EXECUTE LAYER 2 SEND COMMANDS BY USING THE MENU

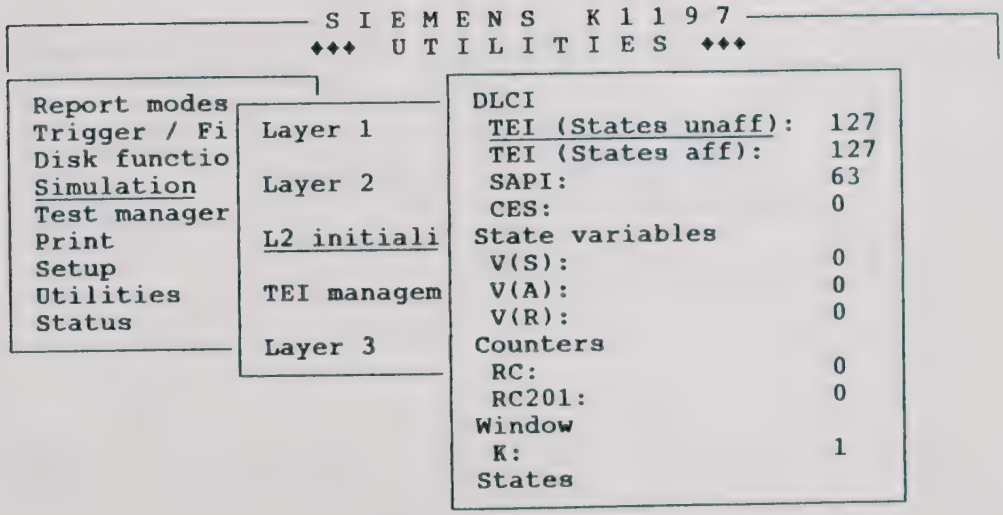



```
ISDN D-CHANNEL EMULATION/SIMULATION Version 2.0  Rev. 0 alpha
OK-7
1 QUIT_MENU 2 HELP 3 COMMAND  4       5       6       7       8
```

TREE: SIMULATE –> LAYER 2

MENU SYSTEM

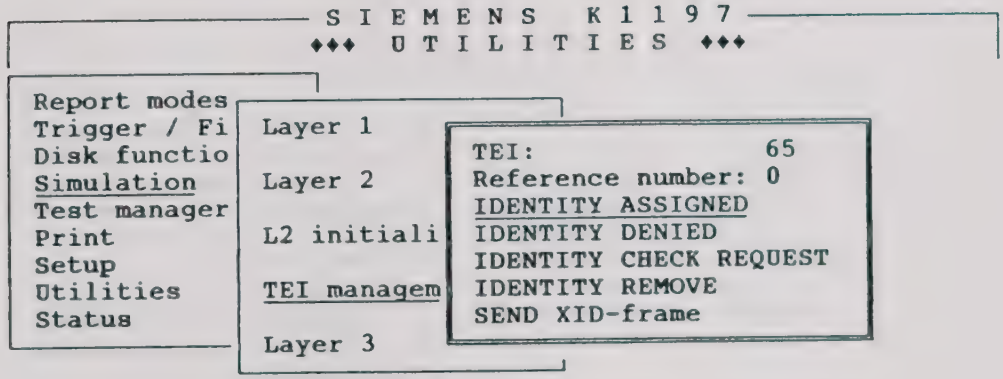
EXAMPLE: SET DATA LINK IDENTIFIERS AND CONTROL VARIABLES BY USING THE MENU



TREE: SIMULATE -> LAYER 2 INITIALIZATION

MENU SYSTEM

EXAMPLE: PERFORM TEI MANAGEMENT FUNCTIONS

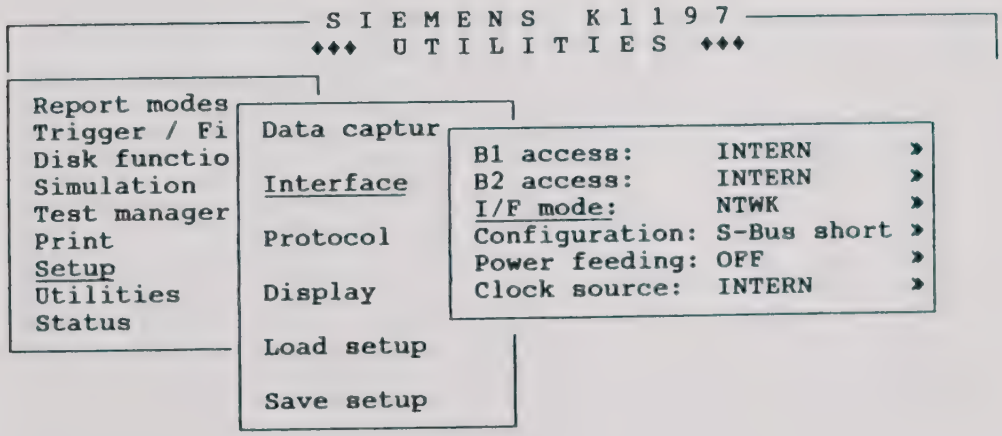


ISDN D-CHANNEL EMULATION/SIMULATION Version 2.0 Rev. 0 alpha
OK-7
1 QUIT_MENU 2 HELP 3 COMMAND 4 5 6 7 8

TREE: SIMULATE -> TEI MANAGEMENT

MENU SYSTEM

EXAMPLE: SET INTERFACE MODE



ISDN D-CHANNEL EMULATION/SIMULATION Version 2.0 Rev. 0 alpha
OK-7
1 QUIT_MENU 2 HELP 3 COMMAND 4 5 6 7 8

TREE: SETUP -> INTERFACE
-> INTERFACE MODE

MENU SYSTEM

EXAMPLE: VERIFY STATUS OF THE 8 DATA LINK CONNECTIONS

SIEMENS K1197
♦♦♦ UTILITIES ♦♦♦

CES	TEI	SAPI	L2ST	RC	K	VR	VS	VA	T200	T203
0	64	0	4	0	1	0	0	0	10	100
1	65	0	70	0	1	0	0	0	10	100
2	127	63	1	0	1	0	0	0	10	100
3	127	63	1	0	1	0	0	0	10	100
4	127	63	1	0	1	0	0	0	10	100
5	127	63	1	0	1	0	0	0	10	100
6	127	63	1	0	1	0	0	0	10	100
7	127	63	1	0	1	0	0	0	10	100

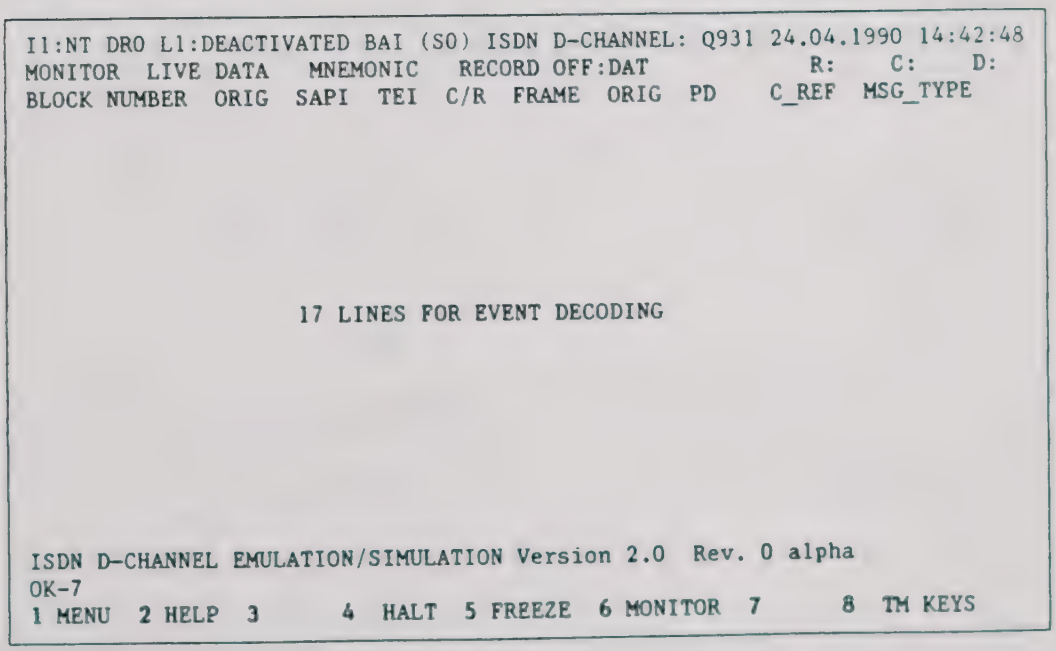
CES	TEI	SAPI	st:L2/ASP	NS	NR	P	F	W	X	Y	Z	T200/201	N200/201/201TX	RC201
2	127	63	1	1	0	0	0	0	0	0	0	10 10	3 260 260	0

ISDN D-CHANNEL EMULATION/SIMULATION Version 2.0 Rev. 0 alpha
OK-7
1 QUIT_MENU 2 HELP 3 COMMAND 4 5 6 7 8

TREE: STATUS -> CONNECTIONS

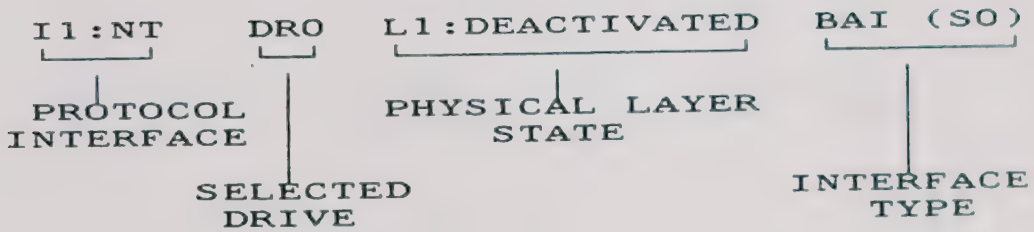
SCREEN LAYOUT (DATA WINDOW ACTIVE)

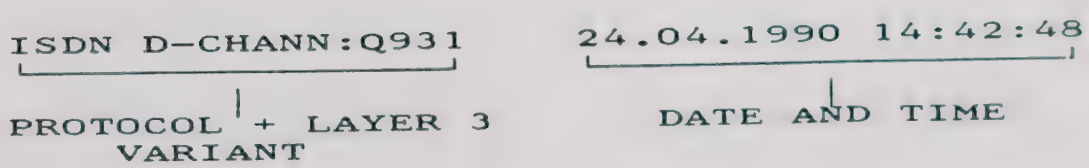
OVERVIEW



SCREEN LAYOUT (DATA WINDOW ACTIVE)

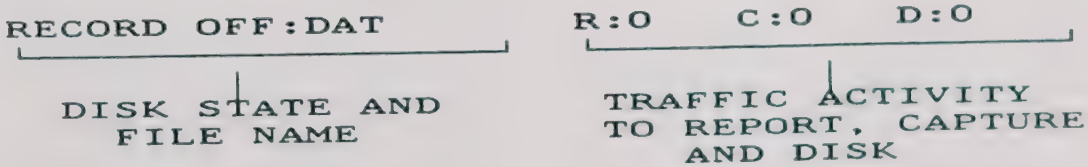
FIRST STATUS LINE





SCREEN LAYOUT (DATA WINDOW ACTIVE)

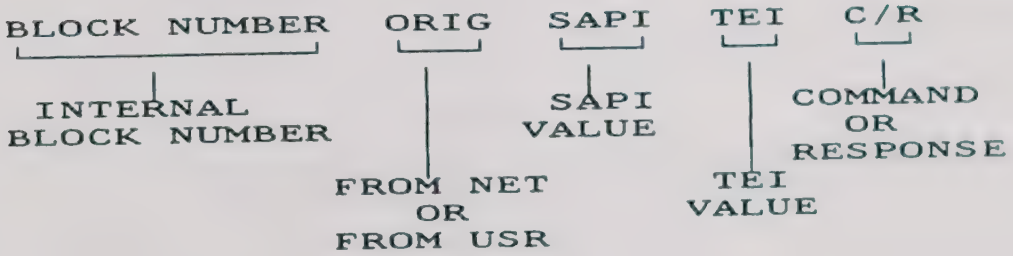
SECOND STATUS LINE

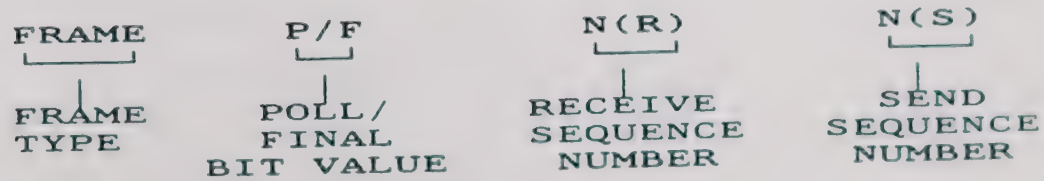


SCREEN LAYOUT (DATA WINDOW ACTIVE)

THIRD STATUS LINE

EVENT RELATED INFORMATION





COMMANDS

ALL CONMMANDS CAN BE EXECUTED BY MENU FUNCTIONS TOO

GENERAL COMMANDS:

- **CLEAR_CRT**

  CLEAR DISPLAY SCREEN

- **PRINT_ON / PRINT_OFF** (HALT MODE)

  ENABLE / DISABLE REPORT PRINTING

- **CAPT_ON / CAPT_OFF**

  SAVE DATA IN CAPTURE RAM WHILE MONITORING / CAPTURING OFF

- **CAPT_WRAP / CAPT_FULL**

  OVERWRITE CAPTURE BUFFER / CAPTURE DATA UNTIL BUFFER IS FULL

- **FROM_CAPT / FROM_DISK** (HALT MODE)

  PLAYBACK FROM CAPTURE / PLAYBACK FROM DISK

- **CTOD_ON / CTOD_OFF** (HALT MODE)

  TRANSFER FROM CAPTURE TO DISK ON/ NO TRANSFER FROM CAPTURE TO DISK

REPORT COMMANDS

- **REP_ON**

  TURN ON REPORT DISPLAY

- **REP_OFF**

  TURN OFF REPORT DISPLAY

- **REP_SHORT**

  REPORT EVENTS IN A SHORT FORMAT (IF REPORT DISPLAY IS TURNED ON)

- **REP_HEX**

  REPORT ALL OCTETS IN HEX FORMAT (IF REPORT DISPLAY IS TURNED ON)

- **REP_CHAR**

  REPORT OCTETS CONTAINING PRINTABLE CHARACTERS (ACCORDING TO THE SELECTED CHARACTER SET) AS CHARACTERS AND REPORT UNPRINTABLE CHARACTERS IN A HEX FORMAT

- **REP_COMP**

  REPORT EVENTS IN MOST COMPLETE FORM

DATA LINK SELECTION COMMANDS

- **CES** (N --)
  SELECT CURRENT CONNECTION ENDPOINT SUFFIX (N = 0 .. 7)

- **CES=0** (--)
  SELECT CES 0 TO BE CURRENT CONNECTION ENDPOINT SUFFIX

- **CES=1..7** (--)
  SIMILAR TO CES=0 FOR THE OTHER CONNECTION ENDPOINT SUFFIXES

- **SAPI** (N --)
  SET SAPI VALUE OF CURRENTLY SELECTED CES

- **TEI** (N --)
  SET TEI VALUE OF CURRENTLY SELECTED CES

- **ATEI** (N --)
  SET TEI VALUE OF CURRENTLY SELECTED CES AND FORCE L2 AND MANAGEMENT STATES TO "TEI ASSIGNED" STATES

- **NEW_TEI** (--)
  SET TEI VALUE OF CURRENTLY SELECTED CES TO 127. TEI ASSIGNMENT PROCEDURE WILL BE EXECUTED WITH NEXT LINK SETUP

FIRST SEND COMMANDS

THE FOLLOWING SEND COMMANDS OPERATE UNDER THE CONTROL OF THE DATA LINK LAYER STATE MACHINES. THE CONNECTION ENDPOINT IS THE "CURRENT CES" AND THE CURRENT VALUES OF SAPI AND TEI ARE EMPLOYED

- **SABME**

  SEND SABME, UA IS EXPECTED. ON THE USER SIDE: IF NO TEI IS CURRENTLY ASSIGNED FOR THE SELECTED CES, TEI ASSIGNMENT WILL BE REQUESTED FROM THE NETWORK FIRST

- **DISC**

  DISC IS SENT IF THE CURRENT CES HAS A TEI ASSSIGNED; UA OR DM IS THEN EXPECTED. WHEN THE UA IS RECEIVED, THE EMULATION ENTERS THE IDENTY ASSIGNED STATE (LAYER 2 STATE 4)

- **DM**

  IF A TEI IS ASSIGNED FOR THE CURRENT CES, A DM FRAME IS TRANSMITTED AND THE EMULATION IS FORCED INTO THE IDENTY ASSIGNED STATE

SERVICE PRIMITIVES

ALL SERVICE PRIMITIVES ARE EXECUTED UNDER THE CONTROL OF THE LAYER 2 STATE MACHINES

- **DL_ESTABLISH**

  ESTABLISH DATA LINK CONNECTION, ASSIGN TEI BEFORE IF NECESSARY

- **DL_RELEASE**

  RELEASE AN ESTABLISHED DATA LINK CONNECTION

- **DL_SET_BUSY**

  SET DATA LINK CONNECTION IN "OWN RECEIVER BUSY" CONDITION

- **DL_CLEAR_BUSY**

  CLEAR A PREVIOUSLY SET "OWN RECEIVER BUSY" CONDITION

DATA SERVICE PRIMITIVES

SENDP **(ADDR --) OR (--)**

SEND LAYER 3 PACKET USING ACKNOWLEDGED OPERATION. BUFFER AT ADDR IN STRING FORMAT (FIRST BYTE IS COUNT BYTE)

DL_DATA **(ADDR --) OR (--)**

SEND LAYER 3 PACKET USING ACKNOWLEDGED OPERATION (SAME AS SENDP)

DL_UNIT_DATA **(ADDR --) OR (--)**

SEND LAYER 3 PACKET USING UNACKNOWLEDGED OPERATION

DL_DATA_REQ **(ADDR\ LENGTH --)**

SIMILAR TO SENDP, BUT LENGTH MAY BE LONGER THAN 255 BYTES

DL_UNIT_DATA_REQ **(ADDR\ LENGTH --)**

SIMILAR TO DL_UNIT_DATA, BUT LENGTH MAY BE LONGER THAN 255 BYTES

MDL_UNIT_DATA **(ADDR --) OR (--)**

SEND A MANAGEMENT/ASP PACKET USING UNACKNOWLEDGED OPERATION. BUFFER AT ADDR IN STRING FORMAT

SEND DATA BUFFERS

THE FOLLOWING TYPES OF DATA BUFFERS FOR SENDING ARE POSSIBLE:

- KEYBOARD STRING ENTRIES (LITERAL BUFFERS PROVIDED BY " ..." OR X" ...")

- USER DEFINED BUFFERS DEFINED BY VARIABLE/ALLOT

- LAYER 3 MESSAGE BUFFERS ACCESSED BY " BUFFER NAME" ?MESSAGE. (THE LENGTH OF THESE BUFFERS IS CONTAINED IN THE FIRST 16 BITS).

- 8 PREDEFINED DATA BUFFERS BEING FILLED BY
  " STRING" MAKE_DATA1
  OR MAKE_DATA2 .. MAKE_DATA8

  AVAILABLE SEND COMMANDS FOR THE PREDEFINED BUFFERS:

  - SENDP/DL_DATA

  - DL_UNIT_DATA

  - MDL_UNIT_DATA

SEND DATA BUFFERS

USAGE OF THE PREDEFINED BUFFERS:

- FILL THE BUFFER(S) USING THE MAKE_DATAn COMMAND,

  e.g. X" 01035A" MAKE_DATA3

- SELECT THE BUFFER TO BE TRANSMITTED, USING THE PKT COMMAND,

  e.g. 3 PKT

- TRANSMIT THE BUFFER USING AN AVAILABLE COMMAND,

  e.g. SENDP

TRANSMIT STRINGS ENTERED AT THE KEYBOARD DIRECTLY:

- SELECT DIRECT TRANSMISSION:

  0 PKT

- ENTER STRING AT THE KEYBOARD,

  e.g. X" 01021A"

- TRANSMIT THE STRING USING AN AVAILABLE COMMAND,

  e.g. SENDP